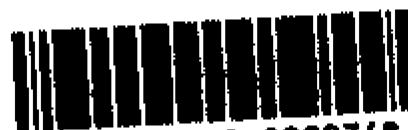
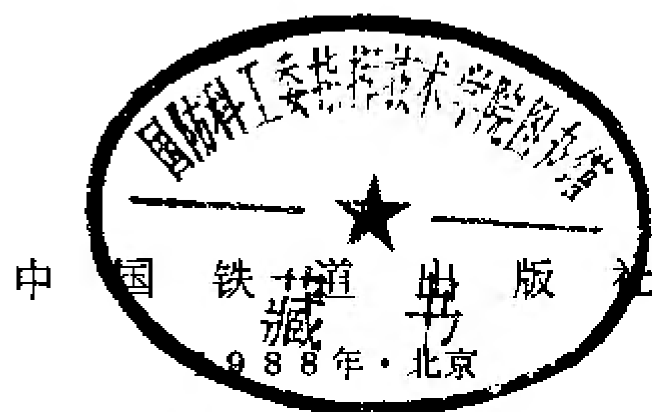
图论常用算法选编

杨洪　编





中国铁道出版社
1988年·北京

内 容 简 介

本书是一部在计算机上使用的有关图论最优化算法的工具书，是用目前国内各种类型电子计算机普遍使用的FORTRAN-Ⅳ书写的。书内汇编了图论中十分有用的一些通用算法，以标准子程序的形式给出了程序清单。

全书共分七部分：图的基本算法，树的计算，图的中心和路的计算，独立集与支配集的计算，匹配算法，着色问题算法，以及网络流的计算等。

读者对象：工程技术人员、大专院校师生。

图论常用算法选编

杨 洪 编

中国铁道出版社出版

责任编辑 黄 燕 林瑞耕 封面设计 安 宏

新华书店总店科技发行所发行

各地新华书店经售

中国铁道出版社印刷厂印

开本：787×1092毫米 1/16 印张：7.75 字数：176千

1988年2月 第1版 第1次印刷

印数：0001—4,000册 定价：1.70元

序

现代科学技术在若干方面都有很大的发展。电子计算机及微型电子计算机的出现及其日益广泛的应用，便是其中之一。从本世纪三、四十年代以来，数学逐渐广泛地渗透到社会的各个领域。由于许多应用问题的需要，产生了一些新的理论和方法，其中图论及其应用是在六十年代以后才迅速发展起来的。图论和电子计算机的使用和发展，在一些方面是互相促进的。

人们在使用电子计算机和微型电子计算机来解应用问题时，需要有针对问题的解法和相应的计算机程序。有了程序，一般说来，只要学会使用就行了。但事实是实际问题的情况往往是复杂的。这时，人们就不仅要学会使用一些程序，而且还需要能灵活应用一些基本的程序，才可能较好地解决较复杂或情况有变化时的问题。于是，按一定系统来编写一些较常用算法的计算机程序就很有必要了。

杨洪同志的《图论常用算法选编》就是这样的一本书。书中既有解一些常见问题的计算机程序（如解最短径路问题，选址问题和匹配问题等的算法的计算机程序），又对图论与网络理论中的重要算法作了介绍，具有一定的系统性，并且在各个算法中作者都举有精选的和验算过的例题，以供读者参考。

谢　力　同

1985年12月

选择了部分基础的、实用性强的算法。本书介绍的全部算法程序都已在PDP-11电子计算机或APPLE-Ⅱ电子计算机上调试通过。因为这些程序都是按照FORTRAN-Ⅳ文本的规定书写的，读者在其它类型电子计算机上运行这些程序时，只需要按照所用机种的规定改变一下输入输出语句的格式，其它内容一般不必改动。

本书编写过程中，得到谢力同教授的亲切关怀，刘家壮、张天锡同志的热情帮助，在此表示衷心的感谢。

由于作者水平所限，书中不妥之处在所难免，欢迎读者批评指正。

杨　洪

1985年11月

目　录

1. 图的基本计算

1.1 图的连通性的计算

1.1.1 功　能

如果已知图的关联矩阵，需要由该矩阵判断图的连通性（此图分为几个连通块）。在本段给出一个简便的方法。

1.1.2 方 法 概 述

已知某图的关联矩阵为$B(n, m)$。其中n为该图顶点总数，m为该图所含弧的总数。计算目的：求出该图的连通块数，并指出这n个顶点分别属于哪一个连通块。将连通块简数目存放于变量S当中，将各顶点所属的连通块的序号存放在数组$Q(n)$当中。

为了便于了解计算的过程，图 1.1 给出程序的流程图。

1.1.3 子程序参数说明

子程序名称　LEING(N,M,B,Q,S)

N：图的顶点数目，整型变量。

M：图的弧总数，整型变量。

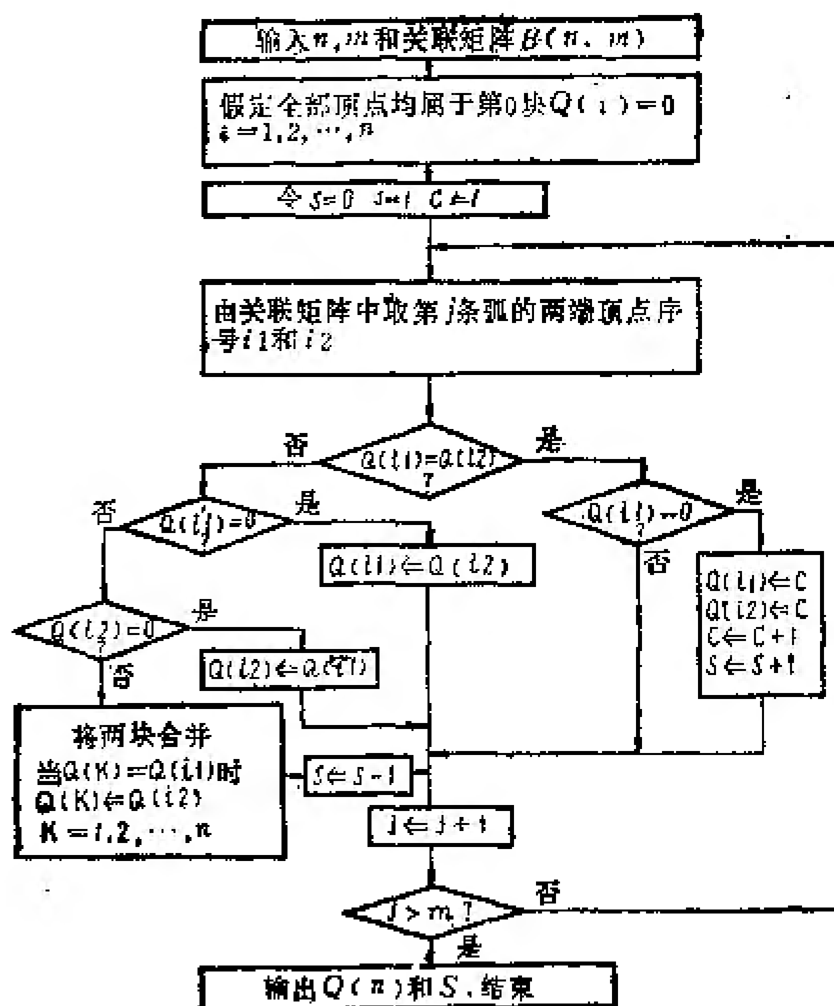


图 1.1

B(N，M)：图的关联矩阵，输入数据，整型数组。其中：

$$B(i,j)=\begin{cases}1, & \text{当第 } j \text{ 弧与第 } i \text{ 顶点关联；}\\ 0, & \text{当第 } j \text{ 弧与第 } i \text{ 顶点不关联；}\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots,M$。

Q(N)，各顶点所属子块的序号，输出结果，$1\leqslant Q(i)\leqslant S$，$i=1,2,\cdots,N$。整型数组。

S：该图所含连通子块的总数，输出结果，整型变量。

1.1.4 子程序

```
      SUBROUTINE LEING(N,M,B,Q,S)
      INTEGER B(N,M), Q(N), L(40), P(2), S, C,D, E
      C=1
      S=0
      DO 1 I=1,N
1     Q(I)=0
      DO 10 J=1,M
      D=1
      DO 3 I=1,N
      IF(B(I,J).NE.1)GOTO 3
      P(D)=I
      D=D+1
3     CONTINUE
      I1=P(1)
      I2=P(2)
      E=Q(I1)
      D=Q(I2)
      IF(E.NE.D)GOTO 5
      IF(E.EQ.0)GOTO 4
      L(J)=E
      GOTO 10
4     L(J)=C
      Q(I1)=C
      Q(I2)=C
      S=S+1
      C=C+1
      GOTO 10
5     IF(E.NE.0)GOTO 6
```

```
      L(J)=Q(I2)
      Q(I1)=Q(I2)
      GOTO 10
6     IF(D.NE.0) GOTO 7
      L(J)=Q(I1)
      Q(I2)=Q(I1)
      GOTO 10
7     DO 8 K=1,N
      IF(Q(K).EQ.E) Q(K)=D
8     CONTINUE
      DO 9 K=1,J
      IF(L(K).EQ.E) L(K)=D
9     S=S-1
10    CONTINUE
      RETURN
      END
```

1.1.5 例　题

1）对以下三个关联矩阵进行连通性计算，求它们的S与$Q(N)$。

（1）$N=10$，$M=6$

$$B(I,J)=\left|\begin{array}{cccccc}
1&0&0&0&0&0\\
0&1&0&0&0&0\\
0&0&1&0&0&0\\
0&0&0&1&1&0\\
0&0&0&0&0&1\\
1&0&0&0&0&0\\
0&1&1&0&0&0\\
0&0&0&1&0&0\\
0&0&0&0&1&0\\
0&0&0&0&0&1
\end{array}\right|$$

（2）$N=10$，$M=11$

$$B(I,J)=\left|\begin{array}{ccccccccccc}
1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&1&1&1&0\\
0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&1&1
\end{array}\right|$$

（3）$N=10$，$M=11$

$$B(I,J)\quad\begin{vmatrix}
1&1&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0\\
0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&0&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0\\
0&0&0&1&0&0&0&0&1&1&0\\
0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&1&0&0&0&0&0&0
\end{vmatrix}$$

2）计算程序

```
      INTEGER B(20,30),Q(20)
      WRITE(7,1)
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)M
1     FORMAT(1X,'N=')
2     FORMAT(1X,'M=')
      CALL LEIN(N,M,B,Q)
      STOP
      END
C
      SUBROUTINE LEIN(N,M,B,Q)
      INTEGER B(N,M),Q(N),S
      WRITE(7,1)
      DO 2 I=1, N
2     READ(5,6) (B(I,J), J=1,M)
6     FORMAT(30I1)
      WRITE(6,1)
```

```
1       FORMAT(1X,'B(I,J)=')
        DO 7 I=1, N
7       WRITE(6,8) (B(I,J),J=1,M)
8       FORMAT(1X,30I3)
        CALL LEING(N,M,B,Q,S)
        WRITE(6,3)S
3       FORMAT(1X,'S=',I3)
        WRITE(6,4) (I,I=1,N)
4       FORMAT(1X,'I=  ',20I3)
        WRITE(6,5) (Q(I),I=1,N)
5       FORMAT(1X,'Q(I)=',20I3)
        RETURN
        END
```

3）计算结果

（1）

```
S=4
I=     1 2 3 4 5 6 7 8 9 10
Q(I) =1 2 2 3 4 1 2 3 3 4
```

（2）

```
S=2
I=     1 2 3 4 5 6 7 8 9 10
Q(I) =1 1 1 1 2 2 2 2 2 2
```

（3）

```
S=2
I-     1 2 3 4 5 6 7 8 9 10
Q(I) =1 1 1 2 2 1 2 1 1 2
```

1.2 求无向图基本回路矩阵的算法

1.2.1 功　能

如果无向连通图$G(V,E)$上存在回路，本段的程序可以求出该图对应于一个生成树的基本回路矩阵。利用本程序的计算结果，可以通过求环和的运算来得到图G的完全回路矩阵。

寻找图的基本回路矩阵的求解方法，对于电路分析等方面是一种有用的方法。

1.2.2 方法概述

已知一无向连通图$G(V,E)$由N个顶点、M条无向弧构成，则基本回路的数目$MC=M-N+1$。对于某一连通图，确定它的一棵生成支撑树是容易的。然而，对于该生成树又可以求出它在图G中的余树。余树由MC条弧组成。生成树与余树的每一条弧形成一条回路，最终得到的MC条回路就是图G的一组基本回路。

求基本回路的计算方法，我们用流程图1.2来表示。

1.2.3 子程序参数说明

子程序名称 CCM(N,M,A,C,MC)

N：图的顶点数目，整型变量。

M：图上所含弧的数目，整型变量。

A(N，M)：图的关联矩阵，输入数据，整型数组。

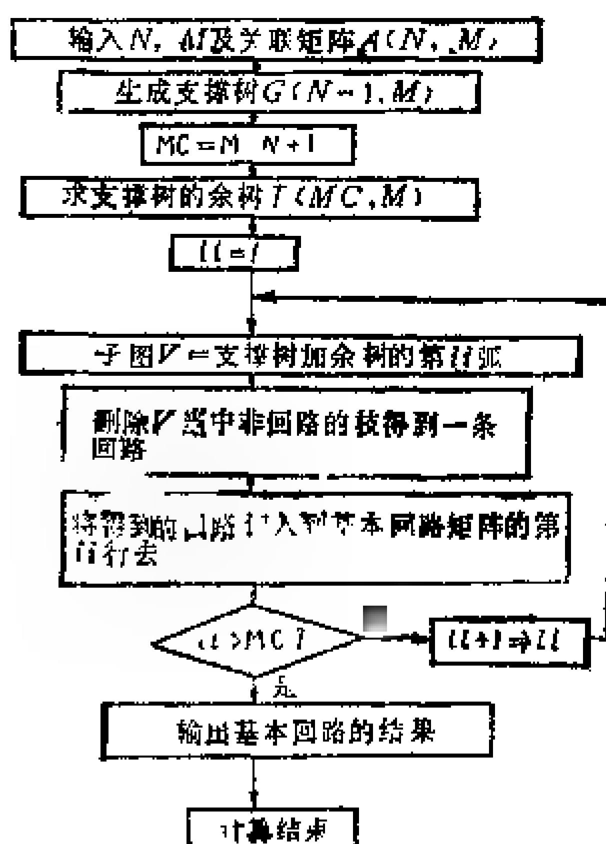


图 1.2

其中：

$$A(i,\ j)=\begin{cases}1，当第 j 弧与第 i 顶点关联；\\ 0，当第 j 弧与第 i 顶点不关联。\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots,M$。

C(30, M)：图的基本回路矩阵，输出结果，整型数组。其中：

$$C(i,\ j)=\begin{cases}1，当第 i 回路包含第 j 弧时；\\ 0，当第 i 回路不包含第 j 弧，或者当 i>MC 时。\end{cases}$$

$i=1,2,\cdots,30$； $j=1,2,\cdots,M$。

MC：图的基本回路数目，整型变量。

1.2.4 子程序

```
          SUBROUTINE CCM(N,M,A,C,MC)
          INTEGER A(N,M), C(30,M), P(50,2),
     #    T(50, 2), V(50,2), G(50,2), S(50)
          N1=N-1
          DO 1 I=1, 50
          DO 1 J=1, 2
          P(I,J)=0
          G(I,J)=0
          T(I,J)=0
1         V(I,J)=0
          DO 3 I=1, M
          K=0
          DO 2 J=1, N
          IF(A(J,I).EQ.0) GOTO 2
          K=K+1
          P(I,K)=J
          T(I,K)=J
          V(I,K)=J
          IF(K.EQ.2) GOTO 3
2         CONTINUE
3         CONTINUE
          J=0
10        DO 4 I=1, M
          IF(V(I,1)×V(I,2).EQ.1.OR.V(I,1),NE.
     #    1.AND.V(I,2).NE.1) GOTO 4
          J=J+1
          G(J,1)=T(I,1)
```

```
      G(J,2)=T(I,2)
      L=MAXO(V(I,1),V(I,2))
      DO 5 II=1, M
      DO 5 JJ=1, 2
      IF(V(II,JJ).EQ.L) V(II,JJ)=1
5     CONTINUE
4     CONTINUE
      IF(J.LT.N1) GOTO 30
      DO 6 I=1, M
      V(I,1)=0
6     V(I,2)=0
      K=0
      DO 26 I=1, M
      DO 25 J=1, N1
      IF(G(J,1).EQ.T(I,1).AND.G(J,2).EQ.
    # T(I,2)) GOTO 26
25    CONTINUE
      K=K+1
      V(K,1)=T(I,1)
      V(K,2)=T(I,2)
26    CONTINUE
      MC=M-N+1
      DO 20 I=1, 30
      DO 20 J=1, M
20    C(I,J)=0
      DO 12 II=1, MC
      DO 8 I=1, N1
      DO 8 J=1, 2
8     T(I,J)=G(I,J)
      T(N,1)=V(II,1)
      T(N,2)=V(II,2)
```

```
      DO 10 J=1,N
10    S(J)=0
      DO 13 I=1, N
      DO 11 J=1, 2
      IF(T(I,J).EQ.0) GOTO 11
      K=T(I,J)
      S(K)=S(K)+1
11    CONTINUE
13    CONTINUE
14    K=0
      DO 16 J=1, N
      IF(S(J).NE.1) GOTO 16
      K=K+1
      DO 15 I=1, M
      IF(T(I,1).NE.J.AND.T(I,2).NE.J)GOTO 15
      L1=T(I,1)
      L2=T(I,2)
      T(I,1)=0
      T(I,2)=0
      S(L1)=S(L1)-1
      S(L2)=S(L2)-1
      GOTO 16
15    CONTINUE
16    CONTINUE
      IF(K.GT.0) GOTO 14
      DO 17 I=1, N
      IF(T(I,1).EQ.0) GOTO 17
      DO 18 J=1, M
      IF(T( ,1).NE.P(J,1)) GOTO 18
      IF(T(I,2).NE.P(J,2)) GOTO 18
      C(II,J)=1
```

```
      GOTO 17
18    CONTINUE
17    CONTINUE
12    CONTINUE
      RETURN
      END
```

1.2.5 例　题

1）求如下两个无向连通图的基本回路矩阵。图上各弧所标的数字，是弧的顺序号。

（1）见图1.3。

$$A(I,\ J)=\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

（2）见图1.4。

$$A(I,J)=\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

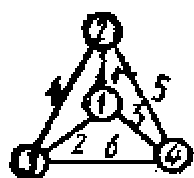


图 1.3

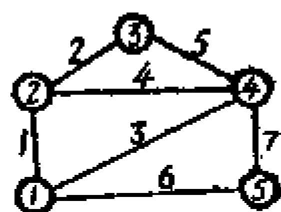


图 1.4

2）计算程序

```
      INTEGER A (50, 50) , C (30, 50)
      WRITE (7,1)
1     FORMAT(1X, 'N=')
      READ(5,3)N
      WRITE(7,2)
      READ(5,3)M
      CALL CC(N,M,A,C)
2     FORMAT(1X,'M=')
3     FORMAT(I3)
      STOP
      END
C
      SUBROUTINE CC(N,M,A,C)
      INTEGER A(N,M),C(30,M)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=')
      DO 2 I=1,N
2     READ(5,3)(A(I,J),J=1,M)
3     FORMAT(50I1)
      CALL CCM(N,M,A,C,MC)
      WRITE(6,6)MC
6     FORMAT(/,1X,'MC=',I3,//,1X,'C(I,J)=')
      DO 7 I=1,MC
7     WRITE(6,8)(C(I,J),J=1,M)
8     FORMAT(1X,50I3)
      RETURN
      END
```

3）计算结果

（1）MC=3

$$C(I,J)=\begin{pmatrix}1&1&0&1&0&0\\1&0&1&0&1&0\\0&1&1&0&0&1\end{pmatrix}$$

见图1.5。

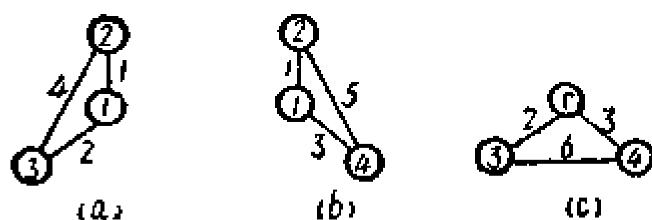


图 1.5

（2）MC＝3

$$C(I,J)=\begin{pmatrix}1&0&1&1&0&0&0\\1&1&1&0&1&0&0\\0&0&1&0&0&1&1\end{pmatrix}$$

见图1.6。

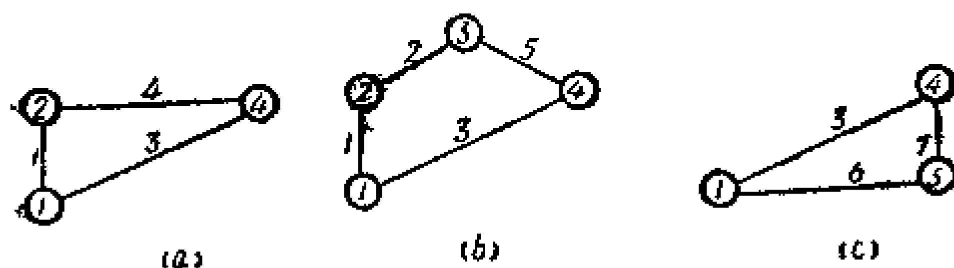


图 1.6

1.3 求有向图基本回路矩阵的算法

1.3.1 功能

本段中处理有向图基本回路矩阵的算法，与前面介绍的

无向图的算法基本上相同。许多电路回路的分析属于有向图的范围。用这一方法不仅能够得到基本回路组，并且在结果中指明了各回路的方向，

1.3.2 方法概述

求有向图基本回路矩阵的算法与无向图中的处理方法基本上相同，在此不作重复说明。为了确定基本回路的方向，在计算过程中的最后一部分增加了回路方向的判别处理。我们规定，回路的方向应当与该回路中属于余树的那一条有向弧的方向一致。在基本回路矩阵中出现的“－1”，表示这条弧的方向与回路方向相反。

1.3.3 子程序参数说明

子程序名称　ECM(N,M,A,C,MC)

N：图的顶点数目，整型变量。

M：图上所含有向弧的数目，整型变量。

A(N,M)：图的关联矩阵，输入数据，整型数组。其中：

$$A(i,j)=\begin{cases}1, & \text{当第 } j \text{ 条弧是第 } i \text{ 顶点的出弧时；}\\ -1, & \text{当第 } j \text{ 条弧是第 } i \text{ 顶点的入弧时；}\\ 0, & \text{当第 } j \text{ 条弧与第 } i \text{ 顶点不关联时。}\end{cases}$$

$$i=1,2,\cdots N;\quad j=1,2,\cdots,M。$$

C(30,M)：图的基本回路矩阵，输出结果，整型数组。其中：

$$C(i,\ j)=\begin{cases}1, & \text{当第 } i \text{ 回路包含第 } j \text{ 弧，且方向一致时；}\\ -1, & \text{当第 } i \text{ 回路包含第 } j \text{ 弧，且方向相反时；}\\ 0, & \text{当第 } i \text{ 回路不包含第} j \text{弧，或 } i>MC \text{时。}\end{cases}$$

$i=1,2,\cdots,30$； $j=1,2,\cdots,M$。

MC：图的基本回路数目，整型变量。

1.3.4 子 程 序

```
      SUBROUTINE ECM(N,M,A,C,MC)
      INTEGER A(N,M),C(30,M),P(50,2),H(50),
     #  T(50,2),V(50,2),G(50,2),S(50)
      N1=N-1
      DO 1 I=1,50
      DO 1 J=1,2
      P(I,J)=0
      G(I,J)=0
      T(I,J)=0
1     V(I,J)=0
      DO 3 I=1,M
      K=0
      DO 2 J=1,N
      IF(A(J,I).EQ.0) GOTO 2
      K=K+1
      P(I,K)=J
      T(I,K)=J
      V(I,K)=J
      IF(K.EQ.2) GOTO 3
2     CONTINUE
3     CONTINUE
      J=0
```

```
30      DO 4 I=1,M
        IF (V(I,1)*V(I,2).EQ.1.OR.V(I,1).NE.
          1.AND.V(I,2).NE.1)GOTO 4
    #   J=J+1
        G(J,1)=T(I,1)
        G(J,2)=T(I,2)
        L=MAX0(V(I,1),V(I,2))
        DO 5 II=1,M
        DO 5 JJ=1,2
        IF(V(II,JJ).EQ.L)V(II,JJ)=1
5       CONTINUE
4       CONTINUE
        IF(J.LT.N1)GOTO 30
        DO 6 I=1,M
        V(I,1)=0
6       V(I,2)=0
        K=0
        DO 26 I=1,M
        DO 25 J=1,N1
        IF(G(J,1).EQ.T(I,1).AND.G(J,2).EQ.
    #   T(I,2))GOTO 26
25      CONTINUE
        K=K+1
        V(K,1)=T(I,1)
        V(K,2)=T(I,2)
        H(K)=I
26      CONTINUE
        MC=M-N+1
        DO 20 I=1,30
        DO 20 J=1,M
20      C(I,J)=0
```

```
      DO 12 II=1,MC
      DO 8 I-1,N1
      DO 8 J=1,2
8     T(I,J)=G(I,J)
      T(N,1)=V(II,1)
      T(N,2)=V(II,2)
      DO 10 J=1,N
10    S(J)=0
      DO 13 I=1,N
      DO 11 J=1,2
      IF(T(I,J).EQ.0) GOTO 11
      K=T(I,J)
      S(K)=S(K)+1
11    CONTINUE
13    CONTINUE
14    K=0
      DO 16 J=1,N
      IF(S(J).NE.1) GOTO 16
      K=K+1
      DO 15 I=1,M
      IF(T(I,1).NE.J.AND.T(I,2).NE.J) GOTO 15
      L1=T(I,1)
      L2=T(I,2)
      T(I,1)=0
      T(I,2)=0
      S(L1)=S(L1)-1
      S(L2)=S(L2)-1
      GOTO 16
15    CONTINUE
16    CONTINUE
      IF(K.GT.0) GOTO 14
```

```
      DO 17 I=1,N
      IF(T(I,1).EQ.0) GOTO 17
      DO 18 J=1,M
      IF(T(I,1).NE.P(J,1)) GOTO 18
      IF(T(I,2).NE.P(J,2)) GOTO 18
      C(II,J)=1
      GOTO 17
18    CONTINUE
17    CONTINUE
      KK=0
      DO 40 I=1,M
      IF(C(II,I).EQ.0.OR.I.EQ.H(II)) GOTO 40
      KK=KK+1
      S(KK)=I
40    CONTINUE
      K=H(II)
      K1=V(II,1)
      K2=V(II,2)
      IF(A(K1,K).EQ.1)K1=K2
      DO 44 J=1,KK
      DO 41 I=1,KK
      K2=S(I)
      IF(K2.EQ.K.OR.A(K1,K2).EQ.0)GOTO 41
      K=K2
      I1=P(K2,1)
      I2=P(K2,2)
      IF(I1.NE.K1)GOTO 42
      K1=I2
      GOTO 43
42    K1=I1
43    IF(A(K1,K2).GT.0)C(II,K2)=-1
```

```
41      CONTINUE
44      CONTINUE
12      CONTINUE
        RETURN
        END
```

1.3.5 例　　题

1）求以下三个有向连通图的基本回路矩阵，并在图上标明基本回路的方向。在下面各例题的图上各有向弧所标的数字是弧的顺序号。

（1）见图1.7。

$N=4$，$M=6$

$$A(I,J)=\begin{pmatrix} -1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 \end{pmatrix}$$

（2）见图1.8。

$N=5$，$M=7$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 1 \end{pmatrix}$$

（3）见图1.9。

$N=4$，$M=5$

$$A(I,J)=\begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 & 1 \end{pmatrix}$$

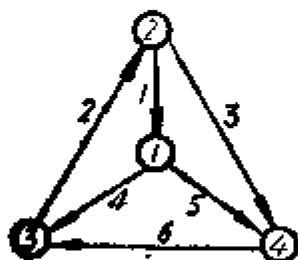


图 1.7

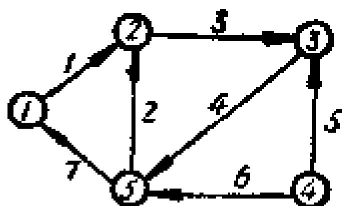


图 1.8

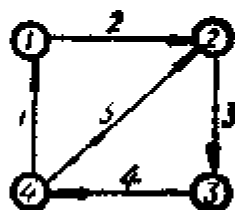


图 1.9

2）计算程序

```
      INTEGER A(50,50),C(30,50)
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,3)N
      WRITE(7,2)
      READ(5,3)M
2     FORMAT(1X,'M=')
3     FORMAT(I3)
      CALL EC(N,M,A,C)
      STOP
      END
C
      SUBROUTINE EC(N,M,A,C)
      INTEGER A(N,M),C(30,M)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=')
      DO 2 I=1,N
```

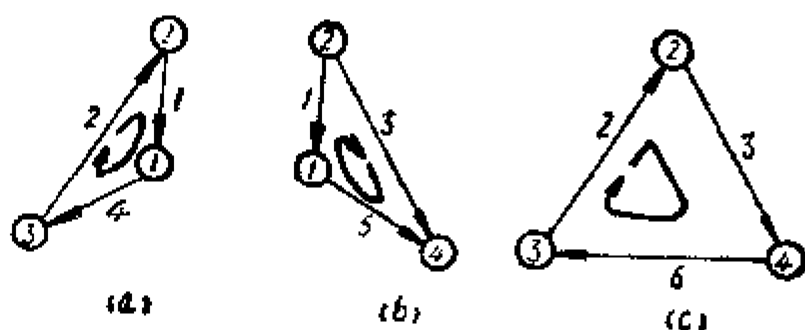
```
2       READ(5,3)(A(I,J),J=1,M)
3       FORMAT(50I3)
        CALL ECM (N,M,A,C,MC)
        WRITE(6,6)MC
6       FORMAT(/,1X,'MC=',I3,//,1X,'C(I,J)=')
        DO 7 I=1,MC
7       WRITE(6,5)(C(I,J),J=1,M)
5       FORMAT(1X,50I3)
        RETURN
        END
```

3）计算结果

(1)MC=3

$$C(I,J)=\begin{pmatrix}1 & 1 & 0 & 1 & 0 & 0\\ 1 & 0 & -1 & 0 & 1 & 0\\ 0 & 1 & 1 & 0 & 0 & 1\end{pmatrix}$$

见图1.10。



(a)　　(b)　　(c)

图 1.10

(2)MC=3

$$C(I,J)=\begin{pmatrix}0 & 1 & 1 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & -1 & 1 & 0\\ 1 & -1 & 0 & 0 & 0 & 0 & 1\end{pmatrix}$$

见图1.11。

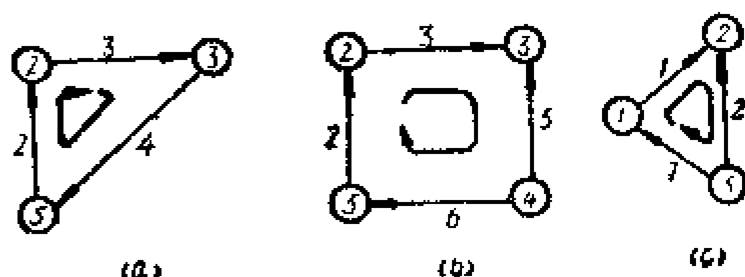


图 1.11

(3) MC - 2

$$C(I,J)=\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

见图1.12

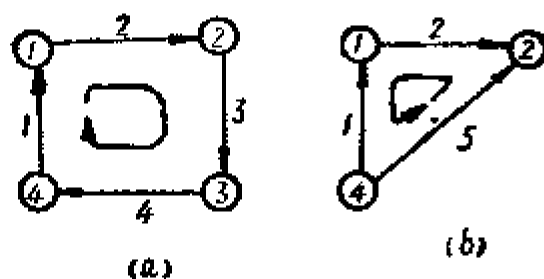


图 1.12

1.4 求无向图基本割集矩阵的算法

1.4.1 功　能

对于无向连通图$G(V,E)$，求它的基本割集矩阵的问题，与求它的基本回路矩阵一样，也是十分有用的。本节给出一个求无向图对应一棵生成树的基本割集矩阵的方法。利用基本割集矩阵，通过环和运算，可以求出该图的完全割集矩阵。

图的割集矩阵，对于电网络分析或交通运输流分析等方面都有实用的价值。

1.4.2 方法概述

一个N顶点、M条弧的无向连通图$G(V,E)$，它的基本割集的数目是$MS-N-1$。本节为求该图的基本割集矩阵采用的方法与前面介绍过的求无向连通图的基本回路矩阵的方法基本上相同。首先，生成该图的一棵支撑树，再从这棵树出发，去确定基本割集矩阵的每个割集应当包括该图上哪些弧。为了方便读者，在此给出一个计算流程图1.13。

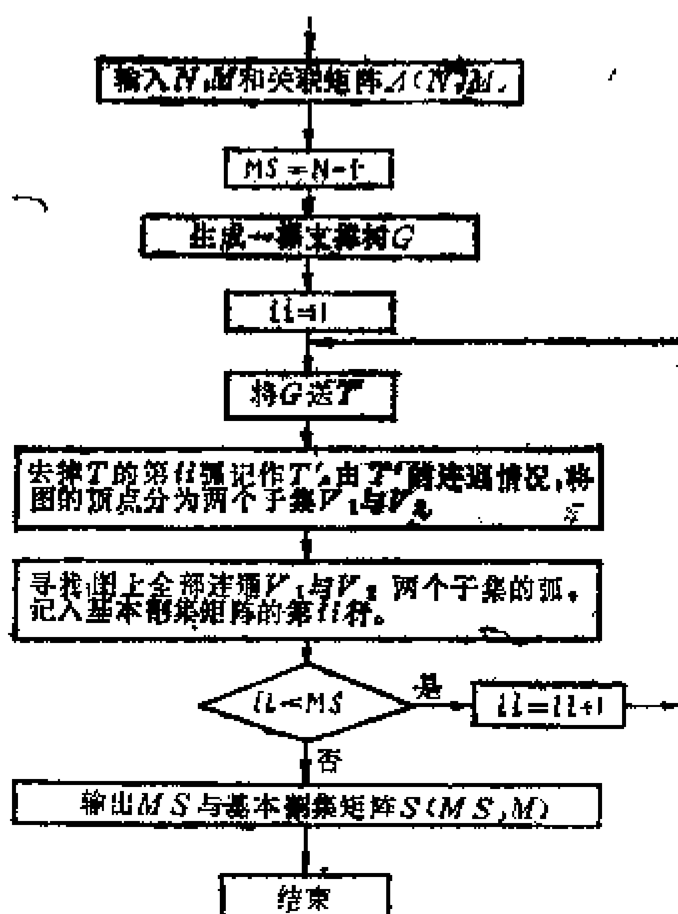


图 1.13

1.4.3 子程序参数说明

子程序名称CSM(N,M,A,S,MS,

N：图的顶点数目，整型变量。

M：图上弧的数目，整型变量。

A(N,M)：图的关联矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1, & \text{当第 } j \text{ 条弧与第 } i \text{ 顶点相关联；}\\ 0, & \text{当第 } j \text{ 条弧与第 } i \text{ 顶点不关联。}\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots,M$。

S(30,M)：图的基本割集矩阵，输出结果，整型数组。

其中：

$$S(i,j)=\begin{cases}1, & \text{当第 } i \text{ 个割集包含第 } j \text{ 条弧；}\\ 0, & \text{当第 } i \text{ 个割集不包含第 } j \text{ 条弧。}\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots,M$。

MS：图的基本割集数目的输出结果，整型变量。

1.4.4 子程序

```
      SUBROUTINE CSM(N,M,A,S,MS)
      INTEGER A(N,M),S(30,M),P(50,2),
#     T(50,2),V(50,2),G(50,2),R(50)
      N1=N-1
      DO 1 I=1,50
      DO 1 J=1,2
      P(I,J)=0
      G(I,J)=0
      T(I,J)=0
```

```
1       V(I,J)=0
        DO 3 I=1,M
        K=0
        DO 2 J=1,N
        IF(A(J,I).EQ.0)GOTO 2
        K=K+1
        P(I,K)=J
        T(I,K)=J
        V(I,K)=J
        IF(K.EQ.2)GOTO 3
2       CONTINUE
3       CONTINUE
        J=0
30      DO 4 I=1,M
        IF(V(I,1)*V(I,2).EQ.1.OR.V(I,1)
     #  .NE.1.AND.V(I,2).NE.1)GOTO 4
        J=J+1
        G(J,1)=T(I,1)
        G(J,2)=T(I,2)
        L=MAX0(V(I,1),V(I,2))
        DO 5 II=1,M
        DO 5 JJ=1,2
        IF(V(II,JJ).EQ.L)V(II,JJ)=1
5       CONTINUE
4       CONTINUE
        IF (J.LT.N1)GOTO 30
        MS=N-1
        DO 20 I=1,30
        DO 20 J=1,M
20      S(I,J)=0
        DO 12 II=1,MS
```

```
        DO 8  I=1,N1
        DO 8  J=1,2
8       T(I,J)=G(I,J)
        T(II,1)=1
        T(II,2)=1
        DO 9  I=1,50
9       R(I)=0
        R(1)=1
        K=1
13      KK=0
        DO 10  I=1,N1
        IF(T(I,1)*T(I,2).EQ.1.OR.T(I,1)
    #   .NE.1.AND.T(I,2).NE.1)GOTO 10
        KK=1
        K=K+1
        IF(T(I,1).NE.1)L=T(I,1)
        IF(T(I,2).NE.1)L=T(I,2)
        R(L)=L
        DO 11  IJ=1,N1
        DO 11  J=1,2
        IF (T(IJ,J).EQ.L)T(IJ,J)=1
11      CONTINUE
10      CONTINUE
        IF(KK.EQ.1)GOTO 13
        DO 14  I=1,M
        I1=P(I,1)
        I2=P(I,2)
        IF(R(I1)+R(I2).EQ.0.OR.R(I1)*R(I2)
    #   .NE.0)GOTO 14
        S(II,I)=1
14      CONTINUE
```

```
12      CONTINUE
        RETURN
        END
```

1.4.5 例　　题

1）求以下两个无向连通图的基本割集矩阵，并在图上用虚线画出这些割集。

（1）见图1.14。$N=4$，$M=6$

$$A(I,J)=\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

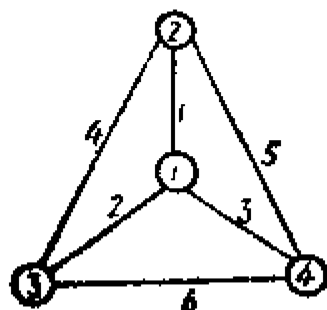


图 1.14

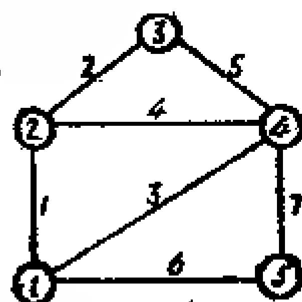


图 1.15

（2）见图1.15。$N=5$，$M=7$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

2）计算程序

```
      INTEGER A(50,50),S(30,50)
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,3)N
      WRITE(7,2)
2     FORMAT(1X,'M=')
      READ(5,3)M
3     FORMAT(I3)
      CALL CS(N,M,A,S)
      STOP
      END
      SUBROUTINE CS(N,M,A,S)
      INTEGER A(N,M),S(30,M)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=')
      DO 2 I=1,N
2     READ(5,3)(A(I,J),J=1,M)
3     FORMAT(50I1)
      WRITE(6,1)
      DO 4 I=1,N
4     WRITE(6,5)(A(I,J),J=1,M)
5     FORMAT(1X,50I3)
      CALL CSM(N,M,A,S,MS)
      WRITE(6,6)MS
6     FORMAT(1X,'MS=',I3,/,1X,'S(I,J)=')
      DO 7 I=1,MS
7     WRITE(6,5)(S(I,J),J=1,M)
      RETURN
      END
```

3）计算结果

(1)MS＝3

$$S(I,J)=\begin{pmatrix}1&0&0&1&1&0\\0&1&0&1&0&1\\0&0&1&0&1&1\end{pmatrix}$$

见图1.16。

(2)MS＝4

$$S(I,J)=\begin{pmatrix}1&0&0&1&1&0&0\\0&1&0&0&1&0&0\\0&0&1&1&1&0&1\\0&0&0&0&0&1&1\end{pmatrix}$$

见图1.17。

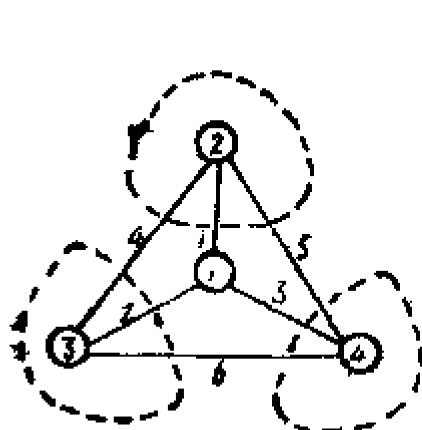


图 1.16

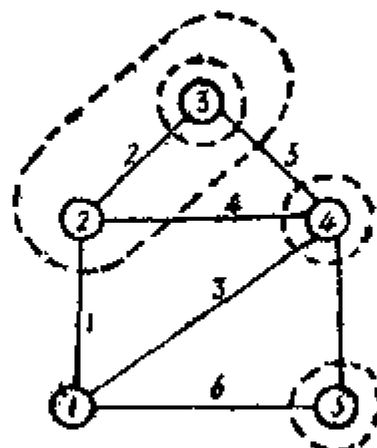


图 1.17

1.5 求有向图基本割集矩阵的算法

1.5.1 功　能

本节的程序与上一节的作用都是求图的基本割集矩阵，而本节的程序适用于有向连通图的情况。

1.5.2 方法概述

本程序所采用的算法与无向图上求基本割集矩阵的方法基本上相同。但是，对有向图规定了割集的方向应当与生成树上属于该割集那一条有向弧的方向一致。因此，在程序中增加了对割集各弧的方向的处理部分。

1.5.3 子程序参数说明

子程序名称ESM(N,M,A,S,MS)

N：图的顶点数目，整型变量。

M：图上弧的数目，整型变量。

A(N,M)：图的关联矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1，当第 j 条弧是第 i 顶点的出弧；\\ -1，当第 j 条弧是第 i 顶点的入弧；\\ 0，当第 j 条弧与第 i 顶点不关联。\end{cases}$$

$i=1,2,\cdots,N；j=1,2,\cdots,M$。

S(30,M)：图的基本割集矩阵，输出结果，整型数组。

其中：

$$S(i,j)=\begin{cases}1，当第 j 条弧属于第 i 个割集，方向与割集方向一致；\\ -1，当第 j 条弧属于第 i 个割集，方向与割集方向相反；\\ 0，当第 j 条弧不属于第 i 个割集。\end{cases}$$

$i=1,2,\cdots,N-1；j=1,2,\cdots,M$。

MS：图的基本割集的数目，整型变量。

1.5.4 子 程 序

```
         SUBROUTINE ESM(N,M,A,S,MS)
         INTEGER A(N,M),S(30,M),P(50,2),
     #   T(50,2),V(50,2),G(50,2),R(50),
     #   H(50)
         N1=N-1
         DO 1 I=1,50
         DO 1 J=1,2
         P(I,J)=0
         G(I,J)=0
         T(I,J)=0
1        V(I,J)=0
         DO 3 I=1,M
         K=0
         DO 2 J=1,N
         IF (A(J,I).EQ.0)GOTO 2
         K=K+1
         P(I,K)=J
         T(I,K)=J
         V(I,K)=J
         IF(K.EQ.2)GOTO 3
2        CONTINUE
3        CONTINUE
         J=0
30       DO 4 I=1,M
         IF (V(I,1)*V(I,2).EQ.1.OR.V(I,1)
     #   .NE.1.AND.V(I,2).NE.1)GOTO 4
         J=J+1
```

```
      G(J,1)=T(I,1)
      G(J,2)=T(I,2)
      H(J)=I
      L=MAX0(V(I,1),V(I,2))
      DO 5 II=1,M
      DO 5 JJ=1,2
      IF(V(II,JJ).EQ.L)V(II,JJ)=1
5     CONTINUE
4     CONTINUE
      IF(J.LT.N1)GOTO 30
      MS=N-1
      DO 20 I=1,30
      DO 20 J=1,M
20    S(I,J)=0
      DO 12 II=1,MS
      DO 8 I=1,N1
      DO 8 J=1,2
8     T(I,J)=G(I,J)
      T(II,1)=1
      T(II,2)=1
      DO 9 I=1,50
9     R(I)=0
      R(1)=1
      K=1
13    KK=0
      DO 10 I=1,N1
      IF(T(I,1)*T(I,2).EQ.1.OR.T(I,1)
   #  .NE.1.AND.T(I,2).NE.1)GOTO 10
      KK=1
      K=K+1
      IF(T(I,1).NE.1)L=T(I,1)
```

```
          IF(T(I,2).NE.1)L=T(I,2)
          R(L)=L
          DO 11 IJ=1,N1
          DO 11 J=1,2
          IF(T(IJ,J).EQ.L)T(IJ,J)=1
11        CONTINUE
10        CONTINUE
          IF(KK.EQ.1)GOTO 13
          K1=G(II,1)
          IF(R(K1).EQ.0)K1=G(II,2)
          K2=H(II)
          K0=A(K1,K2)
          DO 14 I=1,M
          I1=P(I,1)
          I2=P(I,2)
          IF(R(I1)+R(I2).EQ.0.OR.R(I1)*R(I2)
     #    .NE.0)GOTO 14
          S(II,I)=1
          IF(R(I1).NE.0.AND.A(I1,I).NE.K0
     #    .OR.R(I2).NE.0.AND.A(I2,I).NE.
     #    K0)S(II,I)=-1
14        CONTINUE
          IF(S(II,K2).GT.0)GOTO 12
          DO 15 I=1,M
15        S(II,I)=-S(II,I)
12        CONTINUE
          RETURN
          END
```

1.5.5 例　题

1）试求以下二有向连通图的基本割集矩阵。并在图上

用虚线画出这些割集，用箭头标出每个割集的方向。

（1）见图1.18。$N=4$，$M=5$

$$A(I,J)=\begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 & 1 \end{pmatrix}$$

（2）见图1.19。$N=5$，$M=7$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 1 \end{pmatrix}$$

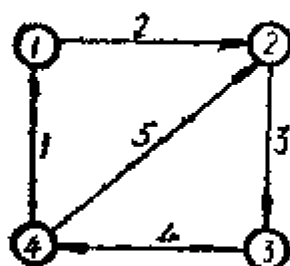


图 1.18

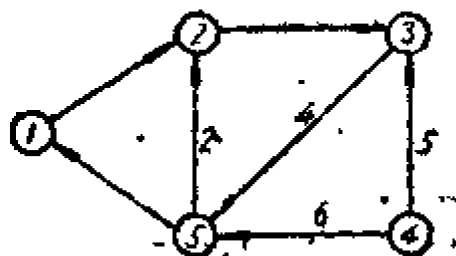


图 1.19

2）计算程序

```
      INTEGER  A(50,50), S(30,50)
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,3)N
      WRITE (7, 2)
      READ (5,3) M
2     FORMAT(1X,'M=')
```

```
3       FORMAT(I3)
        CALL ES(N,M,A,S)
        STOP
        END
C
        SUBROUTINE ES(N,M,A,S)
        INTEGER A(N,M),S(30,M)
        WRITE(7,1)
1       FORMAT(1X,'A(I,J)=')
        DO 2 I=1,N
2       READ(1,3)(A(I,J),J=1,M)
3       FORMAT(50I3)
        WRITE(6,1)
        DO 4 I=1,N
4       WRITE(6,5) (A(I,J),J=1,M)
5       FORMAT(1X,50I3)
        CALL ESM(N,M,A,S,MS)
        WRITE(6,6)MS
6       FORMAT(/,1X,'MS=',I3,//,1X,'S(I,J)')
        DO 7 I=1,MS
7         WRITE(6,5) (S(I,J),J=1,M)
        RETURN
        END
```

3）计算结果

（1）MS＝8

$$S(I,J)=\begin{pmatrix} 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & 0 \end{pmatrix}$$

见图1.20。

（2）MS＝4

$$S(I,J)=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

见图1.21。

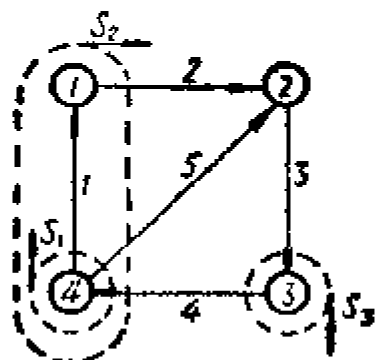


图 1.20

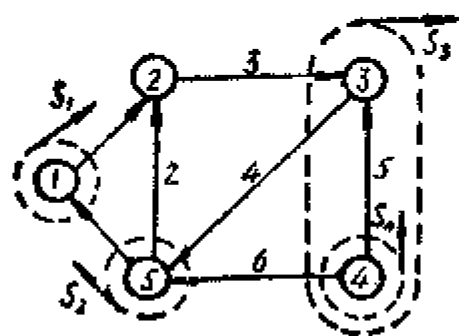


图 1.21

1.6 有向图路径计数的算法 I

1.6.1 功　　能

在给定了n个顶点的有向图 G(V, E) 的 1 路径矩阵 A(n,n) 之后，使用本算法可以求出该图各顶点之间的 1 路径、2 路径、…、直到n路径的数目。从而得到各顶点间的 1 至n路径的总数。

1.6.2 方法概述

已知图$G(V,E)$的 1 路径矩阵$A(n,n)$。其中：

$$A(i,j)=\begin{cases} 1\text{，当顶点}V_i\text{与}V_j\text{存在一条弧}(V_i,V_j)\text{；} \\ 0\text{，当顶点}V_i\text{与}V_j\text{无弧相连。} \end{cases}$$

$i=1,2\cdots,n$; $j=1,2,\cdots,n$。

对于无自圈的图，显然有$A(i,i)=0, i=1,2,\cdots,n$。通过矩阵乘法运算，并用三维数组$B(n,n,n)$的第k层存放k路径的数目，则

$$B(k,n,n)=A^k=A^{k-1}*A \quad k=1,2,\cdots n。$$

而 $S(i,j)=\sum_{k=1}^{n} B(k,i,j) \quad i,j=1,2,\cdots,n$。

存放第i顶点到第j顶点的1至n路径的总数。

1.6.3 子程序参数说明

子程序名称 LYS1(N,A,B,S)

N：图的顶点数目，整型变量。

A(N,N)：图的1路径矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1, & \text{当第}i\text{顶点到第}j\text{顶点有弧相连；}\\ 0, & \text{当第}i\text{顶点到第}j\text{顶点无弧相连。}\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots,N$。

B(N,N,N)：图上各顶点间的1到N路径数目的计算结果。其中B(k,i,j)表示由第i顶点经过k步到达第j顶点的路径数目，整型数组。

S(N,N)：图上各顶点间的全部路径数目的计算结果。其中$S(i,j)$表示由第i顶点到第j顶点的1路径，2路径、…、N路径数目的总和，整型数组。

1.6.4 子程序

```
SUBROUTINE LYS1(N,A,B,S)
```

```
      INTEGER A(N,N),B(N,N,N),H,S(N,N)
      DO 1 I=1,N
      DO 1 J=1,N
1     B(1,I,J)=A(I,J)
      DO 3 K=2,N
      M=K-1
      DO 3 I=1,N
      DO 3 J=1,N
      H=0
      DO 2 L=1,N
2     H=H+B(M,I,L)*A(L,J)
3     B(K,I,J)=H
      DO 6 I=1,N
      DO 6 J=1,N
      H=0
      DO 5 K=1,N
5     H=H+B(K,I,J)
6     S(I,J)=H
      RETURN
      END
```

1.6.5 例 题

1）对以下的五顶点有向图及七顶点有向图，分别求它们的路径计数。我们用S表示起点，用T表示终点。对五顶点图，打印$S=1,T=1$与$S=1,T=5$的1至5路径的数目。对于七顶点图，打印$S=1$，$T=1$与$S=1$，$T=7$的1至7路径数目。

见图1.22。

（1）$N=5$

$$A(N,N)=\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

见图1.23。

（2） $N=7$

$$A(N,N)=\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

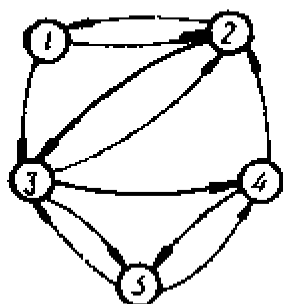


图 1.22

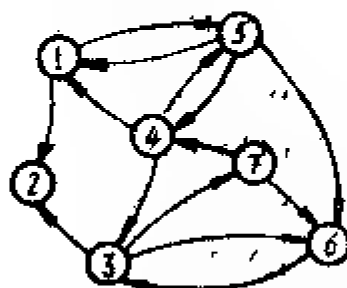


图 1.23

2）计算程序

```
      INTEGER A(15,15), B(15,15,15),S(15,15)
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,2)N
2     FORMAT(I3)
      CALL LY1(N,A,B,S)
      STOP
```

```
C
      END
      SUBROUTINE LY1(N,A,B,S)
      INTEGER A(N,N),B(N,N,N),S(N,N)
      WRITE(7,1)
1     FORMAT(1X,'A(N,N)=',/)
      DO 2 I=1,N
2     READ(5,3) (A(I,J),J=1,N)
3     FORMAT(15I1)
      CALL LYS1(N,A,B,S)
      WRITE(6,7)
      DO 4 I=1,N
4     WRITE(6,5) (S(I,J),J=1,N)
5     FORMAT(1X,15I3)
7     FORMAT(1X,'S(I,J)=',/)
      WRITE(6,8) (B(K,1,1),K=1,N)
8     FORMAT(1X,'S=1 T=1',4X,15I3)
      WRITE(6,9)N,(B(K,1,N),K=1,N)
9     FORMAT(1X,'S=1 T=',I2,4X,15I3)
      RETURN
      END
```

3）计算结果

（1）

$$S(I,J)=\begin{pmatrix} 11 & 25 & 26 & 19 & 19 \\ 12 & 24 & 26 & 19 & 19 \\ 13 & 31 & 30 & 25 & 25 \\ 8 & 23 & 20 & 17 & 18 \\ 11 & 21 & 24 & 19 & 18 \end{pmatrix}$$

```
S=1  T=1      0  1  1  3  6
S=1  T=5      0  1  2  5  11
```

（2）

$$S(I,J)=\begin{pmatrix} 20 & 26 & 30 & 12 & 21 & 38 & 21 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 10 & 0 & 0 & 14 & 8 \\ 35 & 51 & 55 & 20 & 33 & 76 & 44 \\ 33 & 60 & 50 & 21 & 32 & 72 & 42 \\ 0 & 6 & 8 & 0 & 0 & 10 & 6 \\ 0 & 4 & 6 & 0 & 0 & 8 & 4 \end{pmatrix}$$

S=1　T=1　　0　1　1　2　3　5　8

S=1　T=7　　0　0　1　2　2　8　8

1.7 有向图路径计数的算法Ⅱ

1.7.1 说　明

方法Ⅱ与方法Ⅰ基本相同。为了节省存贮单元，在不需要打印各次路径的数目，只需要得到1至N路径的总数的结果时，使用方法Ⅱ为宜。

1.7.2 子程序参数说明

子程序名称LYS 2(N,A,S)

N：图的顶点数目，整型变量。

A(N,N)：图的1路径矩阵，存放方式请看过程$LYS1$的参数说明，整型数组。

S(N,N)：图上各顶点间的全部路径数目的计算结果，整型数组。

1.7.3 子 程 序

```
      SUBROUTINE LYS2(N,A,S)
      INTEGER A(N,N),S(N,N),H(20,20),
     # B(20,20)
      DO 1 I=1,N
      DO 1 J=1,N
      S(I,J)=A(I,J)
    1 B(I,J)=A(I,J)
      DO 5 K=2,N
      DO 3 I=1,N
      DO 3 J=1,N
      H(I,J)=0
      DO 2 L=1,N
    2 H(I,J)=H(I,J)+B(I,L)*A(L,J)
    3 S(I,J)=S(I,J)+H(I,J)
      DO 4 I=1,N
      DO 4 J=1,N
    4 B(I,J)=H(I,J)
    5 CONTINUE
      RETURN
      END
```

1.7.4 例 题

1）用LYS 2 子程序去完成方法 I 的两个例题。

（1） $N=5$

$$A(I,J)=\begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{vmatrix}$$

（2） $N=7$

$$A(I,J)=\begin{vmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

2）计算程序

```
      INTEGER A(20,20),S(20,20)
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,2)N
2     FORMAT(I3)
      CALL LY2(N,A,S)
      STOP
      END
      SUBROUTINE  LY2(N,A,S)
      INTEGER A(N,N),S(N,N)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=',/)
      DO 2  I=1,N
2     READ(5,3) (A(I,J),J=1,N)
3     FORMAT(20I1)
```

```
      CALL LYS2(N,A,S)
      WRITE(6,7)
      DO 4 I=1,N
4     WRITE(6,5) (S(I,J),J=1,N)
5     FORMAT(1X,20I3)
7     FORMAT(1X,'S(I,J)=',/)
      RETURN
      END
```

3）计算结果

（1）

$$S(I,J)=\begin{pmatrix} 11 & 25 & 26 & 19 & 19 \\ 12 & 24 & 25 & 19 & 19 \\ 13 & 31 & 30 & 25 & 25 \\ 8 & 23 & 20 & 17 & 18 \\ 11 & 21 & 24 & 19 & 18 \end{pmatrix}$$

（2）

$$S(I,J)=\begin{pmatrix} 20 & 26 & 30 & 12 & 21 & 38 & 21 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 10 & 0 & 0 & 14 & 8 \\ 33 & 51 & 55 & 20 & 33 & 76 & 44 \\ 33 & 50 & 50 & 21 & 32 & 72 & 42 \\ 0 & 6 & 8 & 0 & 0 & 10 & 6 \\ 0 & 4 & 6 & 0 & 0 & 8 & 4 \end{pmatrix}$$

1.8 道路矩阵的Warshill算法

1.8.1 功　能

在前面，介绍了求N个顶点的有向图 G（V，E）的路径计数的算法。如果我们仅想知道各顶点之间是否有路径相

通，而没有必要去求各次路径的数目，则只需要求出它的道路矩阵$P(N, N)$。本段所提供的计算程序将采用Warshill方法去计算道路矩阵。本方法使用了布尔运算以节省运算的工作量。

1.8.2 方法概述

已知一个N顶点的有向图$G(V, E)$的1路径矩阵$A(N, N)$，则有：

$$S = A + A^2 + A^3 + \cdots + A^N$$

此处用S表示图上各顶点之间的1路径至N路径数目的总和。

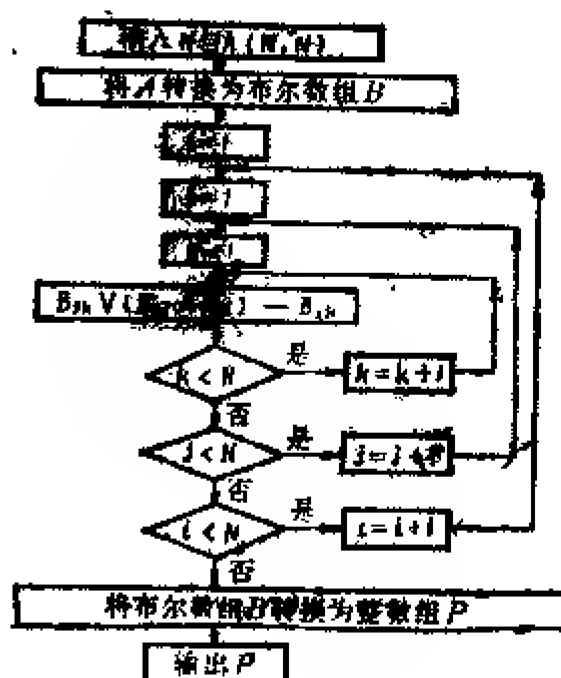


图 1.24

定义：

$$P(i,j) = \begin{cases} 1, & \text{当}S(i,j) \neq 0\text{时；} \\ 0, & \text{当}S(i,j) = 0\text{时。} \end{cases}$$

$$i, j = 1, 2, \cdots, N。$$

称P为图G的道路矩阵。

由A求P的Warshll方法的计算流程图如图1.24。

1.8.3 子程序参数说明

子程序名称　WARSH (N,A,P)

N：图的顶点数目，整型变量。

A(N,N)：图的1路径矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1, & \text{当第}i\text{顶点到第}j\text{顶点有弧相连,}\\0, & \text{当第}i\text{顶点到第}j\text{顶点无弧相连。}\end{cases}$$

$i,j=1,2\cdots,N$。

P(N,N)：图的道路矩阵的计算结果，整型数组。

其中：

$$P(i,j)=\begin{cases}1, & \text{当第}i\text{顶点向第}j\text{顶点有路径相通,}\\0, & \text{当第}i\text{顶点向第}j\text{顶点无路径相通。}\end{cases}$$

$i,j=1,2,\cdots,N$。

1.8.4 子　程　序

```
      SUBROUTINE WARSH(N,A,P)
      INTEGER A(N,N),P(N,N)
      LOGICAL B(20,20)
      DO 1 I=1,N
      DO 1 J=1,N
      P(I,J)=0
1     B(I,J)=A(I,J).EQ.1
      DO 2 I=1,N
      DO 2 J=1,N
```

```
      DO 2 K=1,N
2     B(J,K)=B(J,K).OR.B(J,I).AND.B(I,K)
      DO 3 I=1,N
      DO 3 J=1,N
3     IF(B(I,J))P(I,J)=1
      RETURN
      END
```

1.8.5 例题

1）仍然以有向图路径计数算法 I 的两个例题为例，分别求它们的道路矩阵。

（1）$N=5$

$$A(I,J)=\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

（2）$N=7$

$$A(I,J)=\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

2）计算程序

```
      INTEGER A(20,20),P(20,20)
```

```
      WRITE(7,1)
1     FORMAT(1X,'N=')
      READ(5,2) N
2     FORMAT(I3)
      CALL WA(N,A,P)
      STOP
      END
C
      SUBROUTINE WA(N,A,P)
      INTEGER A(N,N),P(N,N)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=',/)
      DO 2 I=1,N
2     READ(5,3) (A(I,J),J=1,N)
3     FORMAT(20I1)
      CALL WARSH(N,A,P)
      WRITE(6,7)
7     FORMAT(1X,'P(I,J)=')
      DO 4 I=1,N
4     WRITE(6,5) (P(I,J),J=1,N)
5     FORMAT(1X,20I3)
      RETURN
      END
```

3）计算结果

（1）

```
            1  1  1  1  1
            1  1  1  1  1
P(I,J) =    1  1  1  1  1
            1  1  1  1  1
            1  1  1  1  1
```

(2)

$$P(I,J)=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

2. 树 的 计 算

2.1 有向图上树的数目的算法

2.1.1 功　能

在对于有向连通图的研究中，经常需要了解该图包含多少棵树。使用本段的程序，在已知有向图的关联矩阵时，可求出该图包含的全部树的数目。

2.1.2 方 法 概 述

根据 Binet-Cauchy 定理，如果已知某图的基本关联矩阵C，以及它的转置矩阵 C^T，则该图全部树的数目为 det $(C \times C^T)$ 。

本程序的计算步骤如下：

步骤 1　输入图的关联矩阵$A(N,M)$。

步骤 2　取该关联矩阵的前$N-1$行，构成一个基本关联矩阵$C(N-1,M)$。

步骤 3　由矩阵C与它的转置矩阵 C^T 作矩阵乘法运算，得到的结果存放于方阵 $B(N-1,N-1)$ 当中。

步骤 4　求矩阵B的行列式之值，得到 $K=\det(B)$。这里的K就是该图的全部树的数目。

2.1.3 子程序参数说明

1）函数子程序名称KCA(N,M,A)

它是求图上全部树的数目的整型函数子程序。

N：图的顶点数目，整型变量。

M：图上有向弧的数目，整型变量。

A(N,M)：图的关联矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1, & \text{当第}j\text{弧是第}i\text{顶点的出弧；}\\ -1, & \text{当第}j\text{弧是第}i\text{顶点的入弧；}\\ 0, & \text{当第}j\text{弧与第}i\text{顶点不关联。}\end{cases}$$

$i=1,2,\cdots,N$；$j=1,2,\cdots,M$。

2）函数子程序名称DET(N,A)

该子程序是求N阶矩阵A的行列式之值的实型函数子程序。

N：矩阵的阶数，整型变量。

A(N,N)：行列式的输入数据，整型数组。

2.1.4 子 程 序

```
INTEGER FUNCTION KCA(N,M,A)
INTEGER A(N,M)
REAL B(20,20)
N1=N-1
DO 3 I=1,N1
DO 3 J=1,N1
Z=0
```

```
      DO 2 K=M
2     Z=Z+A(I,K)*A(J,K)
3     B(I,J)=Z
      SS=DET(N1,B)
      KCA=SS+0.3
      RETURN
      END
C
      FUNCTION DET(N,A)
      REAL A(20,20)
      NS=1
      PR=1.0
      N1=N-1
      DO 8 K=1,N1
      PM=0.0
      DO 1 I=K,N
      DO 1 J=K,N
      PV=A(I,J)
      IF(ABS(PV).LT.ABS(PM)) GOTO 1
      PM=PV
      I0=I
      J0=J
1     CONTINUE
      IF(PM.EQ.0.0) GOTO 9
      IF(I0.EQ.K) GOTO 3
      NS=-NS
      DO 2 J=K,N
      T=A(I0,J)
      A(I0,J)=A(K,J)
2     A(K,J)=T
3     IF(J0.EQ.K) GOTO 5
```

```
      NS=-NS
      DO 4 I=K,N
      T=A(I,J0)
      A(I,J0)=A(I,K)
4     A(I,K)=T
5     PR=PR*PM
      PM=-1/PM
      K1=K+1
      DO 7 I=K1,N
      Q=A(I,K)*PM
      DO 6 J=K1,N
6     A(I,J)=A(I,J)+Q*A(K,J)
7     CONTINUE
8     CONTINUE
      DET=NS*PR*A(N,N)
      GOTO 10
9     DET=0.0
10    RETURN
      END
```

2.1.5 例 题

1）计算以下三个有向图的全部树的数目。

（1）见图2.1。

$N=4$、$M=6$

$$A(I,J)=\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

（2）见图2.2。

$N=5,\ M=7$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \end{pmatrix}$$

（3）见图2.3。

$N=5,\ M=6$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix}$$

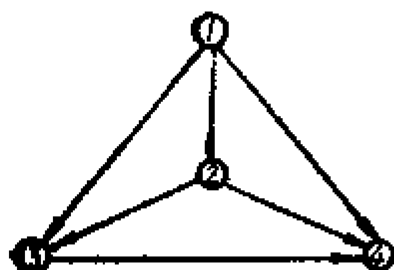


图 2.1

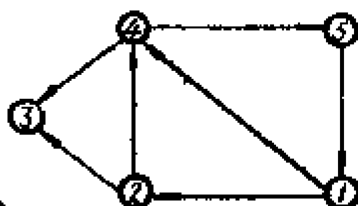


图 2.2

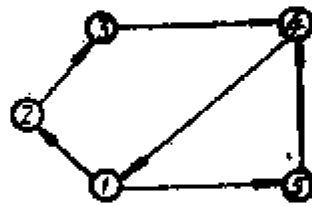


图 2.3

2）计算程序

```
      INTEGER A(20,40)
      WRITE(7.1)
1     FORMAT(1X,'N=')
      READ(5,2)N
2     FORMAT(I3)
      WRITE(7,3)
3     FORMAT(1X,'M=')
      READ(5,2)M
```

```
      CALL  BC(N,M,A)
      STOP
      END
C
      SUBROUTINE BC(N,M,A)
      INTEGER A(N,M)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=',/)
      DO 2 I=1,N
2     READ(5,3)(A(I,J),J=1,M)
3     FORMAT(20I3)
      K=KCA(N,M,A)
      WRITE(6,6)K
6     FORMAT(/,1X,'S=',I3)
      RETURN
      END
```

3）计算结果

（1）S=16

为了便于读者检验结果的正确性，现将这16棵不同的树绘成图2.4。

（2）S=21

（3）S=11

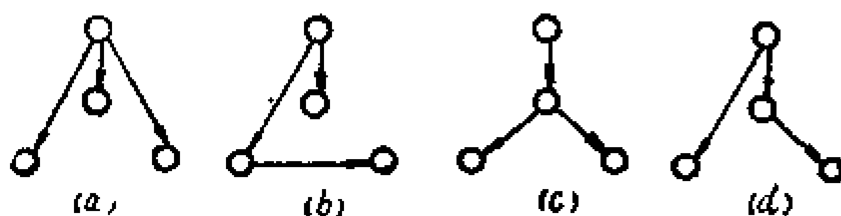


图 2.4之一

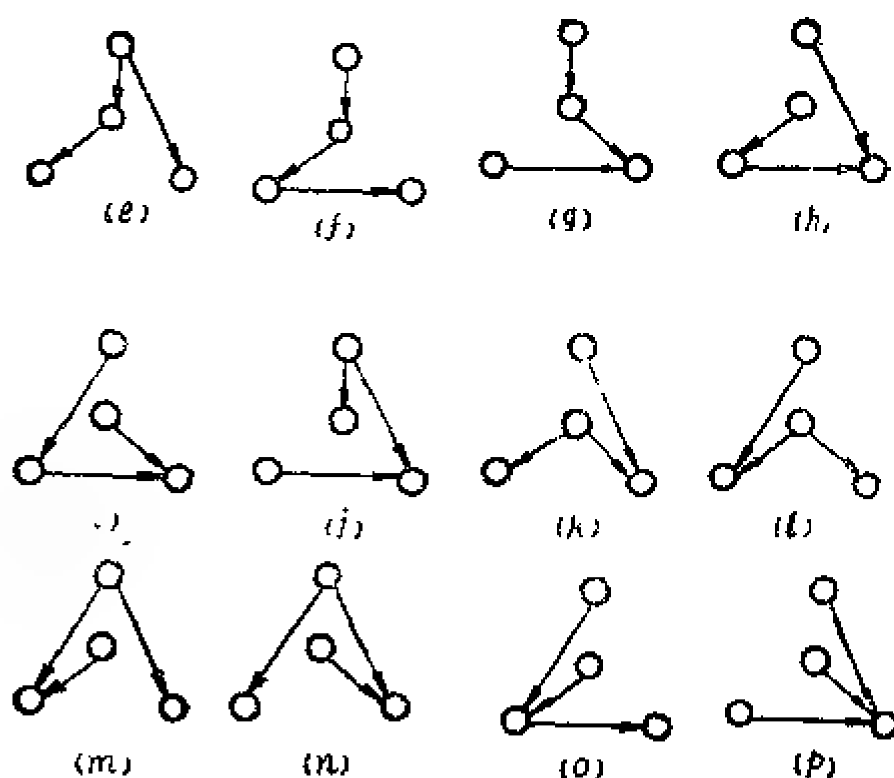


图 2.4

2.2 有向图的外向树与内向树数目的算法

2.2.1 功　能

前面介绍了计算有向图全部树的数目的算法，但未分清其中有几条是外向树，又有几条是内向树，还有几条既非外向树又不是内向树。

所谓外向树，它的特征是除掉一个顶点没有入弧之外，其余所有顶点在这棵树上都有且仅有一条入弧。内向树的特征则相反，除去一个顶点无出弧外，在这棵树上其余所有顶点都有且仅有一条出弧。

本段的程序，可以求出某连通有向图对应一个确定的顶点为起点的全部外向树的数目，并求出以该顶点为终点的全部内向树的数目。

2.2.2 方法概述

已知一个N顶点M条有向弧的连通图G (V,E)。我们的目的是求以K_0号顶点为起点的全部外向树的数目，以及以K_0号顶点为终点的全部内向树的数目。计算步骤如下：

步骤 1　输入图的关联矩阵$A(N, M)$与顶点号K_0。

步骤 2　取矩阵A (N, M) 中K_0行以外的$N-1$行，构成一个基本关联矩阵C ($N-1$, M)。

步骤 3　将矩阵C当中大于0的元素改为0，将C修改之后的全部元素送入D ($N-1$, M) 中。

步骤 4　求$B=C*D^T$，其中B是$N-1$行$N-1$列的方阵，D^T是D的转置矩阵。

步骤 5　求矩阵B行列式的值$K_1=\det(B)$。K_1就是以K_0为起点外向树的数目。

步骤 6　将原基本关联矩阵C ($N-1$, M) 当中大于0的元素不变，小于0的元素改为0，修改后的全部元素送入D ($N-1$, M) 中。

步骤 7　求$B=C*D^T$。

步骤 8　求矩阵B行列式的值$K_2=\det(B)$。K_2就是以K_0为终点内向树的数目。

步骤 9　输出K_1，K_2；计算结束。

2.2.3 子程序参数说明

子程序名称　KCC(N,M,A,K0,K1,K2)

N：图的顶点数目，整型变量。

M：图的有向弧数目，整型变量。

A(N,M)：图的关联矩阵，输入数据，整型数组。

其中：

$$A(i,j)=\begin{cases}1\text{，当第 } j \text{ 弧是第 } i \text{ 顶点的出弧；}\\-1\text{，当第 } j \text{ 弧是第 } i \text{ 顶点的入弧；}\\0\text{，当第 } j \text{ 弧与第 } i \text{ 顶点不关联。}\end{cases}$$

$i=1,2,\cdots,N$； $j=1,2,\cdots M$。

K_0：指定为树的起点与终点的顶点序号，整型变量。

K_1，K_2 ，外向树数目与内向树数目的计算结果，整型变量。

注：子程序中使用了求行列式之使的函数子程序DET(N,A)。它的参数说明与程序清单详见本书前面关于求有向图的全部树的数目的算法一节的有关部分。

2.2.4 子　程　序

```
SUBROUTINE KCC(N,M,A,K0,K1,K2)
INTEGER A(N,M), C(20,40), D(20,40)
REAL B(20,20)
N1=N-1
K=0
DO 2 I=1,N
IF(I.EQ.K0) GOTO 2
```

```
      K=K+1
      DO 1 J=1,M
1     C(K,J)=A(I,J)
2     CONTINUE
      DO 3 I=1, N1
      DO 3 J=1, M
      D(I, J)=C(I, J)
      IF(D(I,J).GT.0)D(I,J)=0
3     CONTINUE
      DO4 I=1, N1
      DO4 J=1, N1
      Z=0.0
      DO 5 K=1, M
5     Z=Z+C(I, K)*D(J.K)
4     B(I, J)=Z
      K1 =(DET(N1 ,B)+0.3)
      DO 6 I=1, N1
      DO 6 J=1, M
      D(I, J)=C(I, J)
      IF(D(I, J) .LT.0) D(I, J)=0
6     CONTINUE
      DO 7 I=1, N1
      DO 7 J=1, N1
      Z=0.0
      DO 8 K=1, M
8     Z=Z+C(I, K)*D(J, K)
7     B(I, J)=Z
      K2 =(DET(N1, B)+0.3)
      RETURN
      END
```

2.2.5 例 题

1）计算以下三个有向图对应每一个顶点的外向树数目与内向树数目。

（1）见图2.5。

$N=4$，$M-6$

$$A(I, J)=\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

（2）见图2.6。

$N=5$，$M-7$

$$A(I, J)=\begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 \end{pmatrix}$$

（3）见图2.7。

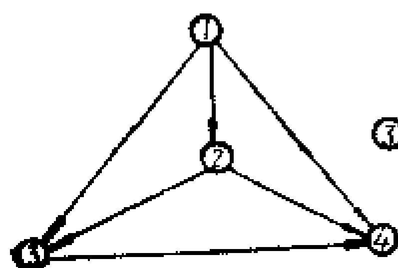


图 2.5

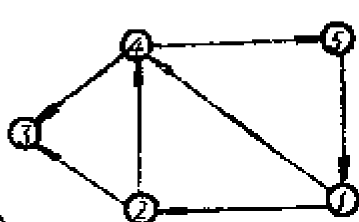


图 2.6

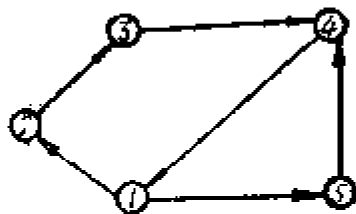


图 2.7

$N=5$, $M=6$

$$A(I, J)=\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix}$$

2）计算程序

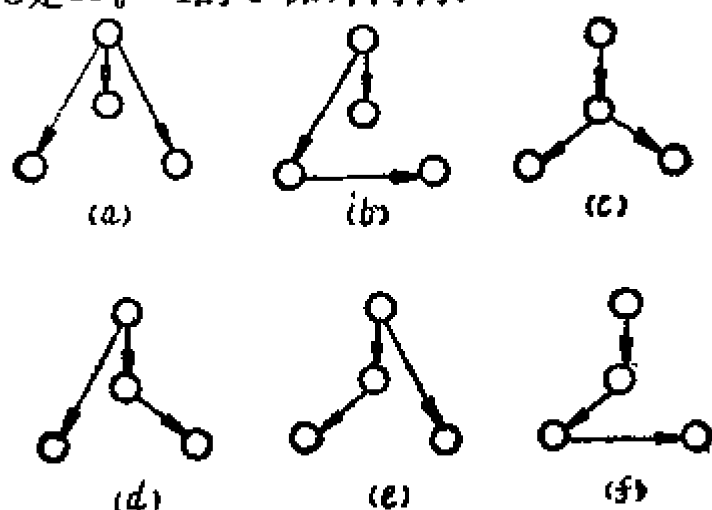
```
      INTEGER A(20, 40)
      WRITE(7, 1)
1     FORMAT(1X, 'N=')
      READ(5, 2)N
2     FORMAT(I3)
      WRITE(7, 3)
3     FORMAT(1X, 'M=')
      CALL BB(N, M, A)
      STOP
      END
C
      SUBROUTINE BB(N, M, A)
      INTEGER A(N,M), K0
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=',/)
      DO 2 I=1,N
2     READ(1,3) (A(I,J), J=1,M)
3     FORMAT(20I3)
      DO 12 K0=1,N
      CALL KCC(N,M,A,K0,K1,K2)
      WRITE(6,6)K0,K1,K2
```

```
6       FORMAT(/,1X,'K0=',I2,4X,'K1=',I2,
     #  1X,'K2=',I2)
12      CONTINUE
        RETURN
        END
```

3）计算结果

（1）K0＝1	K1＝6	K2＝0
K0＝2	K1＝0	K2＝0
K0＝3	K1＝0	K2＝0
K0＝4	K1＝0	K2＝6

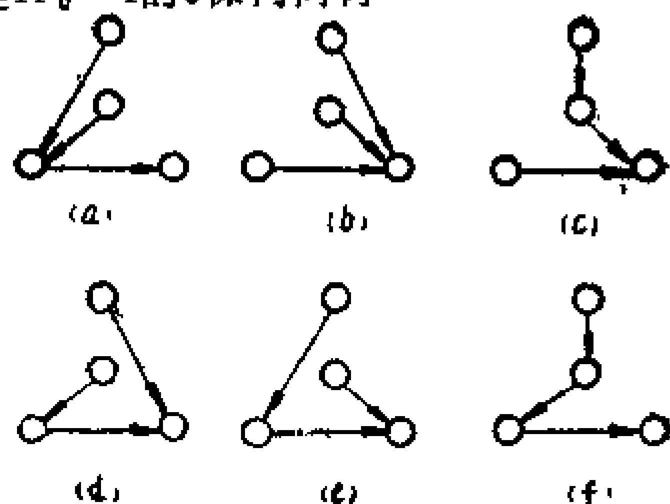
图2.8是K_0　1的6棵外向树：



(a)　(b)　(c)　(d)　(e)　(f)

图　2.8

图2.9是$K_0=4$的6棵内向树



(a)　(b)　(c)　(d)　(e)　(f)

图　2.9

(2) $K_0 = 1$　　$K_1 = 4$　　$K_2 = 0$

　　$K_0 = 2$　　$K_1 = 2$　　$K_2 = 0$

　　$K_0 = 3$　　$K_1 = 0$　　$K_2 = 5$

　　$K_0 = 4$　　$K_1 = 2$　　$K_2 = 0$

　　$K_0 = 5$　　$K_1 = 4$　　$K_2 = 0$

图2.10是$K_0 = 1$的外向树：

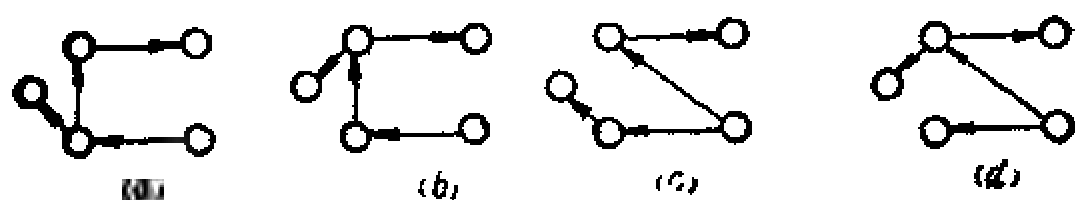


图 2.10

图2.11是$K_0 = 2$的外向树：

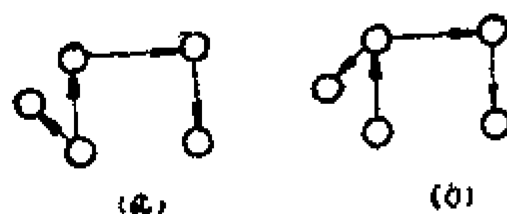


图 2.11

图2.12是$K_0 = 3$内向树：

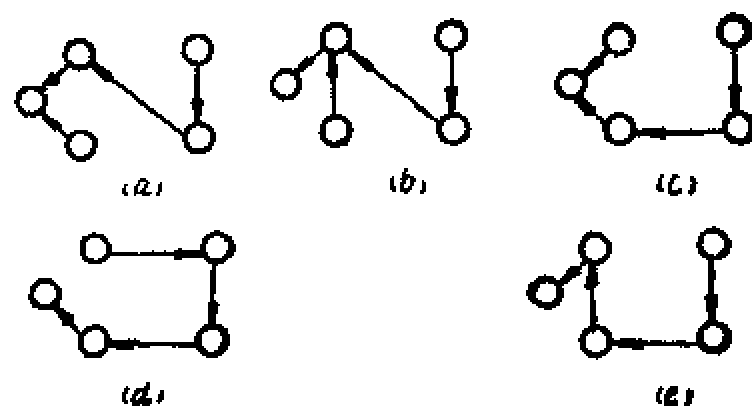


图 2.12

图2.13是$K_0=4$的外向树：

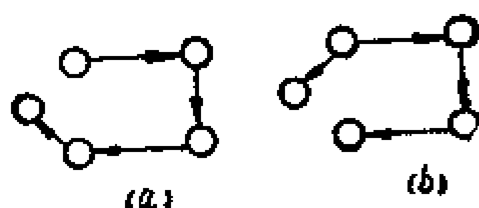


图 2.13

图2.14是$K_0=5$的外向树：

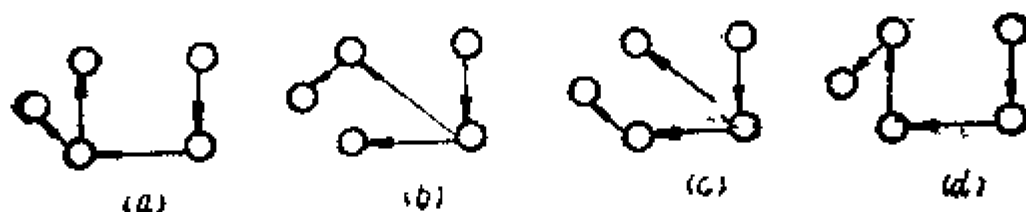


图 2.14

（8）

K0＝1	K1＝2	K2＝1
K0＝2	K1＝1	K2＝1
K0＝3	K1＝1	K2＝1
K0＝4	K1＝1	K2＝2
K0＝5	K1＝1	K2＝1

例（2）的绘图从略。

2.3 无向图最小支撑树的求法

2.3.1 功　能

对于给定的赋权无向连通图，通过本节程序的计算，可

以求出它的一棵总权值最小的支撑树。这棵树连接该图的所有顶点，是该图的一个最小权值连接方案，在生产实践中有许多用途。例如，如何选定一个方案使管道的铺设总长度最短；在交通网的布局方案中，如何选择建设费用最低的方案等，都可以使用最小支撑树这一方法。另外，这一方法也可以用于多元分析的聚类分析方法上去，在生物学、地质学等学科的数值分类学研究中，它是一个有用的工具。

2.3.2 方法概述

已知一个所有的弧都具有大于零的权值的无向连通图，它包括m个顶点向n条弧。由图的连通性可以肯定$n \geqslant m-1$。当$n=m-1$时，图自身就是一棵树，没有讨论的必要。我们只研究$n>m-1$的情况。

将原图的各弧的信息存放在数组$A(n, 3)$当中，其中：

$$\begin{cases} A(i, 1)，存放弧的权值； \\ A(i, 2)，存放弧的一端顶点的顺序号； \\ A(i, 3)，存放弧另一端顶点的顺序号。 \end{cases}$$

$$i=1, 2, \cdots, n。$$

本节的算法采取按照各弧权值的大小，由权值小的弧开始收缩图的顶点，经过$m-1$次收缩，同时记下每次收缩关联的那一条弧的数据，最后得到的就是一棵总权值最小的支撑树的计算结果。

为了计算的方便，首先将数组$A(n, 3)$的数据按照由权值从小到大的顺序重新排序。将重排序之后各弧的顶点序号分别存放到数组C的分量$C(i, 2)$与$C(i, 3)$

当中，这儿 $i=1, 2, \cdots, n$。

收缩步骤如下：

步骤1 $L=0$。

步骤2 选取第一次遇到的 $C(j, 2) \neq C(j, 3)$ 的弧（j 由1开始寻查到 n）。将对应的 $A(j, 1)$，$A(j, 2)$ 和 $A(j, 3)$ 当别记入最小支撑树结果数组 B 的分量当中：

$L=L+1$

$B(L,k)=A(j,k) \qquad k=1,2,3$。

$C(j,2)=C(j,3)=Y=\min(C(j,2), C(j,3))$

$X=\max(C(j,2), C(j,3))$

在数组 C 当中，由 $j+1$ 向 n 寻查，$C(k,2)$ 或 $C(k,3)$ 的值与 X 相等者全部从 Y 值代换（$k=j+1, j+2, \cdots, n$）。

步骤3 如果 $L=m-1$，数组 B 以完全存放了最小支撑树的全部弧，计算结束。如果 $L<m-1$，返回步骤2去，继续寻查与收缩。

2.3.3 子程序参数说明

1）最小支撑树子程序名 KRUSKA(N,M,A,B)

N：图所含弧的条数，整型变量。

M：图所含顶点个数，数型变量 。

A(N,3)：图的每条弧的输入数据，实型数组。

其中：

$A(i, 1)$，存放弧的权值，要求 $A(i, 1)>0$；

$A(i, 2)$，存放弧的一端顶点的顺序号；

$A(i, 3)$，存放弧的另一端顶点的顺序号。

$i=1,2,\cdots,n$。

B(M，3)：存放最小支撑树的输出结果，实型数组，其中：

$B(j,1)$，存放支撑树第 j 条弧的权值；

$B(j,2)$，存放该弧一端顶点的顺序号；

$B(j,3)$，存放该弧另一端顶点的顺序号。

$j=1, 2, \cdots, M-1$。

$B(M,K)$未用到 $(K=1,2,3)$。

2）按列排序子程序名PALL(M,N0,N1,U)

N0：排序数组U的行数，整型变量。

N1：排序数组U的列数，整型变量。

M：指定按其大小排序的列号，要求$1\leqslant M\leqslant$ N1，整型变量。

U(N0,N1)：存放被排序数组的输入数据与输出结果，实型数组。

2.3.4 子 程 序

```
      SUBROUTINE KRUSKA(N,M,A,B)
      DIMENSION A(N,3), B(M,3), C(30,3)
      CALL PALL(1,N,3,A)
      DO 2 J=1, N
      C(J,2)=A(J,2)
2     C(J,3)=A(J,3)
      L=0
      DO 4 J=1, N
      IF(C(J,2) .EQ. C(J,3)) GOTO 1
      X=AMAX1(C(J,2), C(J,3))
```

```
      IF(C(J,2) .LT. C(J,3)) GOTO5
      C(J,2)=C(J,3)
      GOTO6
5     C(J,3)=C(J,2)
6     CONTINUE
      J1=J+1
      DO 8 K=J1,N
      DO 8 I  2,3
      IF (X.EQ.C(K,I)=C(J,2)
8     CONTINUE
      L=L+1
      DO 9 K-1, 3
9     B(L,K)=A(J,K)
1     CONTINUE
      IF(L .EQ. M-1) GOTO 10
4     CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE PALL(M,N0,N1,U)
      DIMENSION U(N0,N1)
      DO 2 I=2, N0
      J=I-1
      J1=0
      IF(U(I,M)-U(J,M)) 1,2,2
4     IF(U(I,M) .GE. U(J,M)) GOTO 3
1     J1=J1+1
      J=J-1
      IF(J .GT. 0) GOTO 4
3     DO 7 I1=1,N1
      X=U(I,I1)
```

```
      J=I
6     J=J-1
      IF(J .LT. I-I1) GOTO 8
      U(J+1,I1)=U(J,I1)
      GOTO 6
8     II=I-I1
7     U(II,I1)=X
2     CONTINUE
      RETURN
      END
```

2.3.5 例　　题

1）求一个10顶点无向赋权图（见图2.15）的最小支撑树。$N=14$，$M=10$。

$$A(i,j)=\begin{pmatrix} 1.0 & 10.0 & 1.0 \\ 2.0 & 1.0 & 3.0 \\ 3.0 & 1.0 & 2.0 \\ 5.0 & 2.0 & 3.0 \\ 6.0 & 3.0 & 4.0 \\ 3.0 & 4.0 & 9.0 \\ 2.0 & 4.0 & 5.0 \\ 2.0 & 2.0 & 5.0 \\ 5.0 & 2.0 & 6.0 \\ 5.0 & 9.0 & 8.0 \\ 4.0 & 5.0 & 8.0 \\ 1.0 & 6.0 & 7.0 \\ 6.0 & 8.0 & 7.0 \\ 6.0 & 5.0 & 6.0 \end{pmatrix}$$

2）计算程序

```
      DIMENSION A(30,3), B(20,3)
      WKITE(7,1)
      READ(5,2) N
      WRITE(7,3)
      READ(5,2)M
1     FORMAT(1X,2HN=)
2     FORMAT(I2)
3     FORMAT(1X,2HM=)
      CALL KRU(N,M,A,B)
      STOP
      END
C
      SUBROUTINE KRU(N,M,A,B)
      DIMENSION A(N,3), B(M,3)
      WRITE(7,10)
10    FORMAT(1X,7HA(I,J)=,/)
      DO 22 I=1,N
22    READ(1,4) (A(I,J), J=1, 3)
4     FORMAT(3F5. 1)
      CALL KRUSKA(N,M,A,B)
      M0=M-1
      WRITE(6,12)
12    FORMAT(1X,7HB(I,J)=,/)
      DO 33 I=1, M0
[illegible] WRITE(6,6) (B(I,K), K=1, 3)
6     FORMAT(4X,F8.1,2F3.0)
      RETURN
      END
```

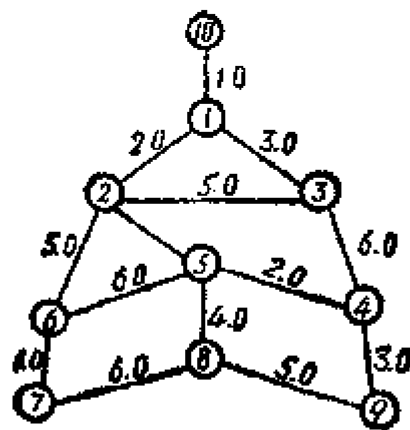


图 2.15

3）计算结果

$$B(i,j)=\begin{bmatrix} 1.0 & 10 & 1 \\ 1.0 & 6 & 7 \\ 2.0 & 1 & 8 \\ 2.0 & 4 & 5 \\ 2.0 & 2 & 5 \\ 3.0 & 1 & 2 \\ 3.0 & 4 & 9 \\ 4.0 & 5 & 8 \\ 5.0 & 2 & 6 \end{bmatrix}$$

见图2.16。

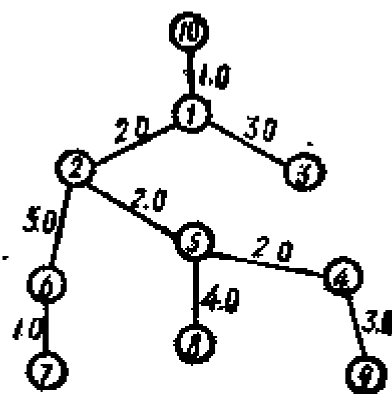


图 2.16

3. 图的中心与路的计算

3.1 连通图中各顶点间最短距离的计算

3.1.1 功　能

已知N顶点无向或有向连通图 $G(V, E)$ 上各顶点间的弧长矩阵。由本节的程序可求出图上所有顶点之间最短通路的路长（也称为顶点之间的最短距离），以最短距离矩阵的形式输出其结果。

求最短距离矩阵的算法，在图论中是一个应用十分广泛的方法。它不仅可以直接用于运输网络的分析上，而且在图上最优选址一类问题中也要使用这种方法首先求出图的最短距离矩阵。本书中有关图的各类中心的算法，都使用了本节的程序。

3.1.2 方法概述

已知n个顶点的有向或无向连通图 $G(V, E)$，它的n个顶点用记号$\{v_1, v_2, \cdots, v_n\}$表示。

首先输入该图的所有顶点之间的弧长矩阵：

$$D=\begin{pmatrix} d_{11} & \cdots & d_{1n} \\ d_{12} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots \\ d_{n} & \cdots & d_{nn} \end{pmatrix},$$

其中：

$$d_{ij}=\begin{cases} v_iv_j\text{的弧长，} & \text{当顶点}v_i\text{至顶点}v_j\text{存在一条弧；}\\ \infty, & \text{当顶点}v_i\text{至顶点}v_j\text{不存在弧；}\\ 0, & \text{当}i=j\text{时。}\end{cases}$$

$$i,\ j=1,\ 2,\ \cdots,\ n。$$

由于$d_{ij}=\infty$时，无穷大的数值是无法记入计算机的存贮单元的。我们可以用一个充分大的实数来替代，只要此数值大于该图上所有弧长的总和就可以了。

另外应当指出，对于无向图显然有$d_{ij}=d_{ji}$　$i,j=1,2,\cdots,n$。处理方法不作任何改变。

为了求出两点之间的最短距离，按照Dijkstra方法，只要反复使用迭代公式$d_{ij}^{(k)}=\min(d_{ij}^{(k-1)},d_{ik}^{(k-1)}+d_{kj}^{(k-1)})$

$$i,j=1,2,\cdots,n;\quad k=1,2,\cdots,n。$$

就可以得到最后的结果。其中，k为迭代次数。$d_{ij}^{(0)}=d_{ij}$是照弧长矩阵的对应分量，作为最短距离矩阵的初值。当$k=n$时，$d_{ij}^{(n)}$就是第v_i顶点至第v_j顶点的最短距离的计算结果。我们把最短距离矩阵的计算结果仍然存放在数组D当中，由它作为输出数据。

3.1.3 子程序参数说明

子程序名称　MINDL0(N,D)

N：图的顶点数目，整型变量。

D(N,N)：图上各顶点间的弧长矩阵。作为子程序的输入数据。当第i顶点至第j顶点无弧邻接时，用一个大于所有弧长之和的常数来代替

∞填入数组的分量$D(i, j)$。计算完成之后，作为输出数据，存放图的最短距离矩阵的计算结果。实型数组。

3.1.4 子 程 序

```
      SUBROUTINE MINDL0(N,D)
      REAL D(N,N), C0(20,20), C(20,20)
      DO 1 I=1, N
      DO 1 J=1, N
1     C0(I,J)=D(I,J)
      DO 6 K=1, N
      DO 4 I=1, N
      DO 4 J=1, N
      X=C0(I,K)+C0(K,J)
4     C(I,J)=AMIN1(X,C0(I,J))
      DO 5 I=1, N
      DO 5 J=1, N
5     C0(I,J)=C(I,J)
6     CONTINUE
      DO 7 I=1, N
      DO 7 J=1, N
7     D(I,J)=C(I,J)
      RETURN
      END
```

3.1.5 例 题

1）计算以下两个无向连通图的最短距离矩阵。在打印输出结果时，我们把最短距离矩阵记作DD。

（1）见图3.1。

$N = 4$

$$D(I,J)=\begin{bmatrix} 0.0 & 1.0 & 3.0 & 4.0 \\ 1.0 & 0.0 & 2.0 & 90.0 \\ 3.0 & 2.0 & 0.0 & 5.0 \\ 4.0 & 90.0 & 5.0 & 0.0 \end{bmatrix}$$

（2）见图3.2

$N = 7$

$D(I,J) =$

$$\begin{bmatrix} 0.0 & 1.0 & 2.0 & 90.0 & 90.0 & 90.0 & 90.0 \\ 1.0 & 0.0 & 1.5 & 0.8 & 3.0 & 90.0 & 90.0 \\ 2.0 & 1.5 & 0.0 & 1.3 & 90.0 & 2.7 & 90.0 \\ 90.0 & 0.8 & 1.3 & 0.0 & 0.5 & 1.4 & 90.0 \\ 90.0 & 3.0 & 90.0 & 0.5 & 0.0 & 1.8 & 2.2 \\ 90.0 & 90.0 & 2.7 & 1.4 & 1.8 & 0.0 & 3.5 \\ 90.0 & 90.0 & 90.0 & 90.0 & 2.2 & 3.5 & 0.0 \end{bmatrix}$$

2）计算程序

```
REAL D(20,20)
WRLTE(7,1)
READ(5,2)N
```

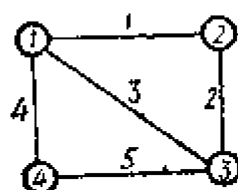


图 3.1

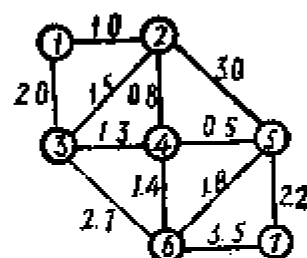


图 3.2

```
1       FORMAT(1X,'N=')
2       FORMAT(I2)
        CALL ZZM(N,D)
        STOP
        END
C
        SUBROUTINE ZZM(N,D)
        REAL D(N,N)
        WRITE(7,2)
2       FORMAT(/,1X,'D(I,J)=',/)
        DO 4 I=1,N
4       READ(5,9) (D(I,J), J=1, N)
9       FORMAT(20F6.1)
        CALL MINDL0(N,D)
        WRITE(6,6)
6       FORMAT(/,1X,'DD(I,J)=',/)
        DO 7 I=1,N
7       WRITE(6,3) (D(I,J), J=1, N)
        RETURN
        END
```

3）计算结果

（1）

$$DD(I,J)=\begin{pmatrix} 0.0 & 1.0 & 3.0 & 4.0 \\ 1.0 & 0.0 & 2.0 & 5.0 \\ 3.0 & 2.0 & 0.0 & 5.0 \\ 4.0 & 5.0 & 5.0 & 0.0 \end{pmatrix}$$

(2)

$$DD(I,J)=\begin{pmatrix} 0.0 & 1.0 & 2.0 & 1.8 & 2.3 & 3.2 & 4.5 \\ 1.0 & 0.0 & 1.5 & 0.8 & 1.3 & 2.2 & 3.5 \\ 2.0 & 1.5 & 0.0 & 1.3 & 1.8 & 2.7 & 4.0 \\ 1.8 & 0.8 & 1.3 & 0.0 & 0.5 & 1.4 & 2.7 \\ 2.3 & 1.3 & 1.8 & 0.5 & 0.0 & 1.8 & 2.2 \\ 3.2 & 2.2 & 2.7 & 1.4 & 1.8 & 0.0 & 3.5 \\ 4.5 & 3.5 & 4.0 & 2.7 & 2.2 & 3.5 & 0.0 \end{pmatrix}$$

3.2 图上两个顶点间最短通路的计算

3.2.1 功　能

对于有向或无向的连通图，任意两个图上的顶点之间是否存在通路？如果存在不只一条通路，哪一条通路是总距离最短的？这条最短通路顺序经过哪些顶点？这是交通运输网络以及其它专业当中时常要解决的问题。本节所提供的方法，用以解决连通图上已确定了出发点为S、终点为T之后，如何找到一条最短程通路的问题。

3.2.2 方法概述

最短路的算法较多，这里采用的是Dijkstra的标号法。

已知图$G(V, E)$的顶点数目为n，各顶点之间的距离矩阵为$A(n,n)$。目的是怎样找到一条由出发点S到终点T的最短通路，给出总路长，以及这条通路顺序经过的全部顶点的标号。为了处理简便，本程序采用了整数运算。在使用中遇到弧长为实数时，可将弧长单位扩大适当的倍数，转换为整型量来处理，也可以将程序中相应的变量和数组改为实型来用，其计算步骤是一样的。

在程序当中，用数组$D(n, 3)$存放图上各顶点的标号信息。其中：

$$D(i, 1)=\begin{cases}1, & \text{当第 } i \text{ 号顶点已标号；}\\ 0, & \text{当第 } i \text{ 号顶点未标号。}\end{cases}$$

$i=1, 2, \cdots, n$。

$D(i, 2)$存放由S点至第i点的最短通路的长度。

$D(i, 3)$存放由S点至第i点的最短通路的前一个顶点的标号。$i=1, 2, \cdots, n$。

Dijkstra算法分以下步骤进行：

步骤1 开始。所有的顶点均未标号，此时$D(i, 1)=0$，$D(i, 2)=M$，$D(i, 3)=S$。$D(S, 1)=1$，$D(S, 2)=0$。这里，M是弧长上界。在图中，如果某两个顶点之间不存在一条弧把它们直接连通，在距离矩阵的相应行列位置原应当以$+\infty$表示之，但计算机上处理问题时，只许存放不超过本机允许范围的数值。我们在程序当中对M取一个比图上所有弧长之和还要大的确定数值。在后面各程序当中用到弧长上界时，均用这一办法处理，以后就不再一一说明了。

令$j=S$，转步骤2。

步骤2 对每一个未标号的顶点i，利用公式$D(i, 2)=\min\limits_{i\text{未标号}}(D(i, 2), D(j, 2)+A(j, i))$修改其距离。

如果对于所有未标号的i都有$D(i, 2)=M$，则说明从S到任何未标号的顶点都没有通路，停止计算。否则在这些$D(i, 2)$当中选一个值最小的，其顶点号记作$G0$，$D(G0, 2)=\min\limits_{i\text{未标号}}\{D(i, 2)\}$。令$D(G0, 1)=1$。当$D(j, 2)+A(j, i)<D(i, 2)$时，令$D(i, 3)=j$。转向步骤3。

步骤3 检查第T号顶点是否已经标号了？如果$D(T, 1)=0$，则将$G0$作为新的起点，返回步骤2，继续上面

的计算。如果$D(T, 1) \neq 0$，则转向步骤4。

步骤4　反向追踪。由$D(i, 3)=T$出发，反向寻查顶点标号，直到$D(i, 3)=S$为止，得到一串由T至S的最短路所经过顶点的反向序号。对这串序号进行反顺序重排，存放到数组$D0(N)$，并在变量$D00$存放这条最短路的总长度。计算全部结束。

3.2.3 子程序参数说明

子程序名称　DIJK11(N,A,D0,D00,S,T,M0,N1)

N：图的顶点数目，整型变量。

A(N,N)：图上各顶点间的原始距离矩阵。输入数据，整型数组。如果第i顶点至第j顶点不存在有向弧，则$A(i,j)=M0$，这里$M0$是弧长上界。对于无向图，有$A(i,j)=A(j,i)$。

D0(N)：由S点至T点的最短路顺序经过的顶点号，输出结果，整型数组。一般情况下这条路的顶点序号不会达到$N-1$个，空余单元存放整数零。

D00：由S点至T点最短通路长度，输出结果，整型变量。

S：最短路的起点，输入数据，整型变量。

T：最短路的终点，输入数据，整型变量。

M0：弧长上界，给一个比所有弧长之和更大的整数，整型变量。

NN：由S至T最短通路所经过的顶点数目，整型变量。

3.2.4 子 程 序

```
      SUBROUTINE DIJK11 (N,A,D0,D00, S, T,#
      M0, NN)
      INTEGER S, T, D00, A(N,N), D0(N)
   #  D(20,3), W, V, U, G, G0
      D00=0
      DO 1 I=1, N
      D0(I)=0
      D(I,1)=0
      D(I,2)=M0
1     D(I,3)=S
      D(S,1)=1
      D(S,2)=0
      J=S
2     G=M0
      G0=0
      DO 3 I=1. N
      IF(D(I,1) .NE. 0) GOTO 3
      U=D(I,2)
      V=D(J,2)+A(J,I)
      W=MIN0(U,V)
      D(I,2)=W
      IF(W .GE. M0) GOTO 3
      IF(U .GT. V)  D(I,3)=J
      IF(W .GE. G) GOTO 3
      G0=I
      G=W
3     CONTINUE
      IF(G0 .NE. 0) GOTO 4
```

```
      D00=M0
      GOTO 12
4     D(G0,1)=1
      J=G0
      IF(D(T,1) .EQ. 0) GOTO 2
      D00=D(T,2)
      D0(1)=T
      J=T
      K=2
      IF(D00 .EQ. M0) GOTO 12
5     J=D(J,3)
      D0(K)=J
      K=K+1
      IF(J .NE. S) GOTO 5
      NN=K-1
      IF(NN/2 .EQ. NN/2.0) NM=NN/2
      IF(NN/2 .NE. NN/2.0) NM=(NN-1)/2
      DO 22 K=1, NM
      KK=D0(K)
      JJ=NN-K+1
      D0(K)=D0(JJ)
22    D0(JJ)=KK
12    CONTINUE
      RETURN
      END
```

3.2.5 例　　题

1）计算以下两个图上各顶点间的最短通路。这里规定弧长上界取900。

（1） 6个顶点的有向图如图3.3。

$N=6$, $M0=900$

$$A(I,J)=\begin{pmatrix} 0 & 1 & 900 & 2 & 900 & 6 \\ 2 & 0 & 4 & 900 & 900 & 900 \\ 900 & 3 & 0 & 900 & 900 & 2 \\ 900 & 1 & 900 & 0 & 3 & 900 \\ 900 & 900 & 900 & 2 & 0 & 1 \\ 900 & 900 & 3 & 900 & 1 & 0 \end{pmatrix}$$

（2） 8个顶点的无向图如图3.4。

$N=8$, $M0=900$

$A(I,J)=$

$$\begin{pmatrix} 0 & 2 & 4 & 5 & 900 & 900 & 900 & 900 \\ 2 & 0 & 1 & 900 & 900 & 900 & 6 & 900 \\ 4 & 1 & 0 & 4 & 3 & 900 & 900 & 900 \\ 5 & 900 & 4 & 0 & 900 & 6 & 900 & 900 \\ 900 & 900 & 3 & 900 & 0 & 2 & 3 & 4 \\ 900 & 900 & 900 & 6 & 2 & 0 & 900 & 1 \\ 900 & 6 & 900 & 900 & 3 & 900 & 0 & 7 \\ 900 & 900 & 900 & 900 & 4 & 1 & 7 & 0 \end{pmatrix}$$

2）计算程序

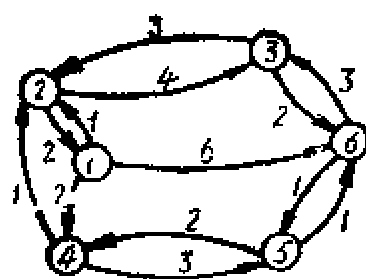


图 3.3

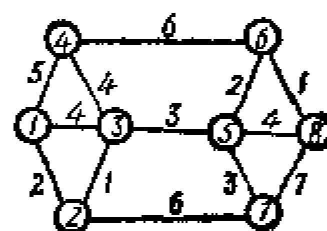


图 3.4

```
          INTEGER A(20, 20), D0(20)
          WRITE(7, 20)
          READ(5,21)N
          WRITE(7,22)
          READ(5,21)M0
          CALL DIJ(N,M0,A,D0)
20        FORMAT(1X,2HN=)
21        FORMAT(I2)
22        FORMAT(1X,3HM0=)
          STOP
          END
C
          SUBROUTINE DIJ(N,M0, A,D0)
          INTEGER S,T,D00, A(N,N),D0(N)
          WRITE(7,1)N,M0
1         FORMAT(1X,'N=',I3,'M0=',I4,//,1X,
     #    'A(I,J)=',/)
          DO 24 I=1,N
24        READ(5,40)(A(I,J),J=1,N)
40        FORMAT(20F8.2)
```

```
      DO 26 S=1,N
      DO 26 T=1,N
      IF(S.EQ.T)GOTO 26
      CALL DIJK11(N,A,D0,D00,S,T,M0,NN)
      WRITE(6,31)S,T,D00,(D0(I),I=1,NN)
31    FORMAT(1X,2HS=,I2,1X,2HT=,I2,1X,
   #  2HD=,I4,4H…,20I3)
26    CONTINUE
      RETURN
      END
```

3）计算结果

（1）分别见图3.5～图3.10。

```
S=1  T=2  D=  1…1 2
S=1  T=3  D=  5…1 2 3
S=1  T=4  D=  2…1 4
S=1  T=5  D=  5…1 4 5
S=1  T=6  D=  6…1 6
S=2  T=1  D=  2…2 1
S=2  T=3  D=  4…2 3
S=2  T=4  D=  4…2 1 4
S=2  T=5  D=  7…2 1 4 5
S=2  T=6  D=  6…2 3 6
S=3  T=1  D=  5…3 2 1
S=3  T=2  D=  3…3 2
S=3  T=4  D=  5…3 6 5 4
S=3  T=5  D=  3…3 6 5
```

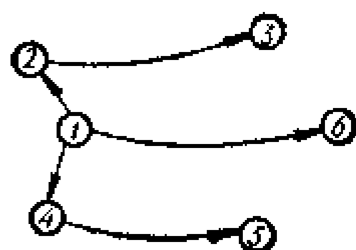


图 3.5

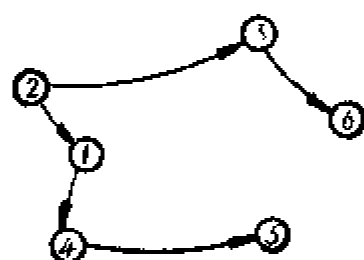


图 3.6

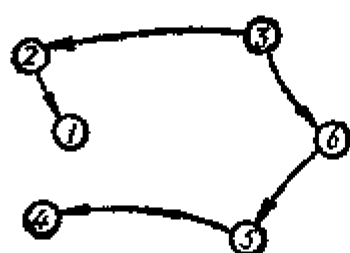


图 3.7

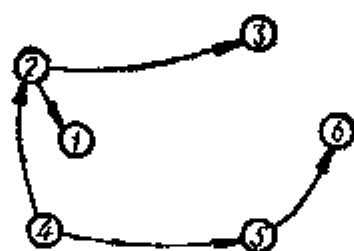


图 3.8

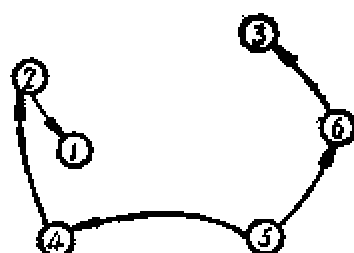


图 3.9

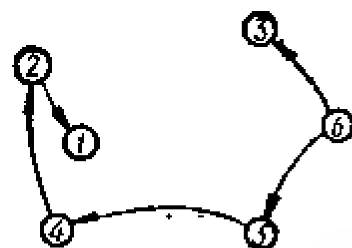


图 3.10

S＝3　T＝6　D＝ 2…3 6

S＝4　T＝1　D＝ 3…4 2 1

S＝4　T＝2　D＝ 1…4 2

S＝4　T＝3　D＝ 5…4 2 3

S＝4　T＝5　D＝ 3…4 5

S＝4　T＝6　D＝ 4…4 5 6

S＝5　T＝1　D＝ 5…5 4 2 1

S＝5　T＝2　D＝ 3…5 4 2

S＝5　T＝3　D＝ 4…5 6 3

S＝5　T＝4　D＝ 2…5 4

S＝5　T＝6　D＝ 1…5 6

S＝6　T＝1　D＝ 6…6 5 4 2 1

S＝6　T＝2　D＝ 4…6 5 4 2

S＝6　T＝3　D＝ 3…6 3

S＝6　T＝4　D＝ 3…6 5 4

S＝6　T＝5　D＝ 1…6 5

（2）分别见图3.11～图3.18。

S＝1　T＝2　D＝ 2…1 2

S＝1　T＝3　D＝ 3…1 2 3

S＝1　T＝4　D＝ 5…1 4

S＝1　T＝5　D＝ 6…1 2 3 5

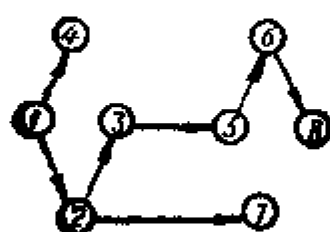


图 3.11

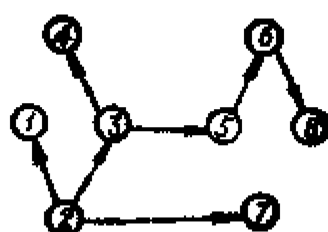


图 3.12

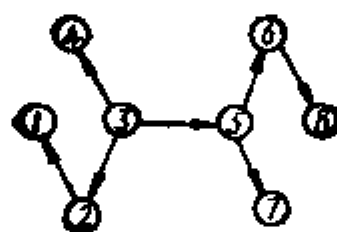


图 3.13

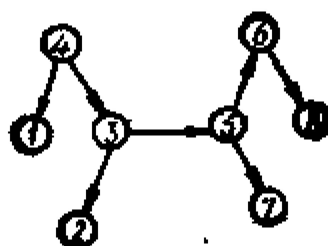


图 3.14

S＝1　T＝6　D＝ 8…1 2 3 5 6

S＝1　T＝7　D＝ 8…1 2 7

S=1 T=8 D= 9…1 2 3 5 6 8
S=2 T=1 D= 2…2 1
S=2 T=3 D= 1…2 3
S=2 T=4 D= 5…2 3 4
S=2 T=5 D= 4…2 3 5
S=2 T=6 D= 6…2 3 5 6
S=2 T=7 D= 6…2 7
S=2 T=8 D= 7…2 3 5 6 8

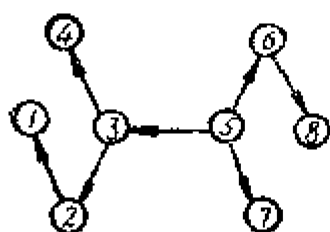


图 3.15

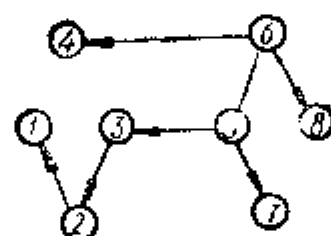


图 3.16

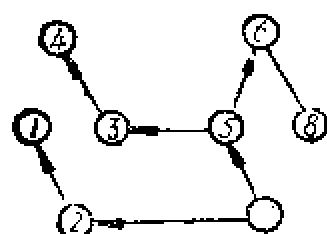


图 3.17

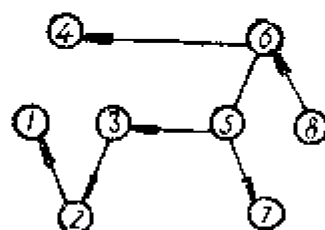


图 3.18

S=3 T=1 D= 3…3 2 1
S=3 T=2 D= 1…3 2
S=3 T=4 D= 4…3 4
S=3 T=5 D= 3…3 5
S=3 T=6 D= 5…3 5 6
S=3 T=7 D= 6…3 5 7
S=3 T=8 D= 6…3 5 6 8
S=4 T=1 D= 5…4 1

S＝4　T＝2　D＝ 5…4 3 2
S＝4　T＝3　D＝ 4…4 3
S＝4　T＝5　D＝ 7…4 3 5
S＝4　T＝6　D＝ 6…4 6
S＝4　T＝7　D＝10…4 3 5 7
S＝4　T＝8　D＝ 7…4 6 8
S＝5　T＝1　D＝ 6…5 3 2 1
S＝5　T＝2　D＝ 4…5 3 2
S＝5　T＝3　D＝ 3…5 3
S＝5　T＝4　D＝ 7…5 3 4
S＝5　T＝6　D＝ 2…5 6
S＝5　T＝7　D＝ 3…5 7
S＝5　T＝8　D＝ 3…5 6 8
S＝6　T＝1　D＝ 8…6 5 3 2 1
S＝6　T＝2　D＝ 6…6 5 3 2
S＝6　T＝3　D＝ 5…6 5 3
S＝6　T＝4　D＝ 6…6 4
S＝6　T＝5　D＝ 2…6 5
S＝6　T＝7　D＝ 5…6 5 7
S＝6　T＝8　D＝ 1…6 8
S＝7　T＝1　D＝ 8…7 2 1
S＝7　T＝2　D＝ 6…7 2
S＝7　T＝3　D＝ 6…7 5 3
S＝7　T＝4　D＝10…7 5 3 4
S＝7　T＝5　D＝ 3…7 5
S＝7　T＝6　D＝ 5…7 5 6
S＝7　T＝8　D＝ 6…7 5 6 8
S＝8　T＝1　D＝ 9…8 6 5 3 2 1
S＝8　T＝2　D＝ 7…8 6 5 3 2
S＝8　T＝3　D＝ 6…8 6 5 3
S＝8　T＝4　D＝ 7…8 6 4

```
S=8  T=5  D= 3…8 6 5
S=8  T=6  D= 1…8 6
S=8  T=7  D= 6…8 6 5 7
```

3.3 图的中心和加权中心的算法

3.3.1 功　能

在连通图的选址问题中，最简单的情况是在该图的所有顶点中确定一个顶点，作为该图的中心。其条件是：这个顶点在所有顶点中与离它本身最远顶点的距离取极小值，则称这一顶点为该图的中心。如果图的每一个顶点赋有各自的权值，在考虑到各顶点权值的条件下，选出的中心称为该图的加权中心。本节的两个子程序分别用以求出图的中心与图的加权中心。

求某一连通图的中心与加权中心，对于确定图上的服务点一类问题是十分有用的。

3.3.2 方 法 概 述

对于 n 个顶点的连通图 $G(V, E)$，无论求它的中心或加权中心，首先都要求出该图的最短距离矩阵。然后，求该图的中心或加权中心就十分容易了。

图的中心计算：

在最短距离矩阵中，求各行的极大值：

$$d_i^0=\max_{1\leqslant j\leqslant n}\{d_{ij}\}\quad i=1,2,\cdots,n。$$

在 n 个极大值当中选取最小者：

$$d^0_{i_0}=\min_{1\leqslant i\leqslant n}\{d^0_i\}$$

则第i_0号顶点被确定为该图的中心。

图的加权中心计算：

事先输入每个顶点的权值$A=(a_1,\ a_2,\ \cdots,\ a_n)$。利用前面求出的该图最短距离矩阵的数据，求

$$d^1_i=\sum_{j=1}^{n}a_jd_{ij}\quad i=1,2,\cdots,n。$$

然后，在这n个值中选取最小者：

$$d^1_{i_1}=\min_{1\leqslant i\leqslant n}\{d^1_i\}$$

则第i_1号顶点被确定为该图的加权中心。

3.3.3 子程序参数说明

1）子程序名称MIND1(N, D, D0, K0)

求图的中心子程序名。

N：图的顶点数目，整型变量。

D(*N*, *N*)：图的最短距离矩阵的输入数据，实型数组。

D0：图的中心离它自身最远顶点的最短距离的输出结果，实型简单变量。

K0：图的中心的顶点序号，输出结果，整型简单变量。

2）子程序名称MIND2（N,D,P,D0,K0）

求图的加权中心子程序名。

N：图的顶点数目，整型变量。

D（N, N）：图的最短距离矩阵的输入数据，实型数

组。

P（N）：图上各顶点的权值输入数据，实型数组。

D0：图的加权中心与其它所有顶点的最短距离的加权总和，输出结果，实型变量。

K0：图的加权中心的顶点序号，输出结果，整型变量。

3.3.4 子程序

```
      SUBROUTINE MIND1(N,D,D0,K0)
      REAL D(N,N),D1
      D0=1E30
      DO 2 I=1,N
      D1=0.0
      DO 1 J=1,N
1     D1=AMAX1(D1,D(I,J))
      IF(D0.LE.D1)GOTO2
      D0=D1
      K0=I
2     CONTINUE
      RETURN
      END
CCC
      SUBROUTINE MIND2(N,D,P,D0,K0)
      REAL D(N,N),P(N),D1
      D0=1E30
      DO 2 I=1,N
      D1=0.0
      DO 1 J=1,N
1     D1=D1+P(J)*D(I,J)
```

```
      IF(D0.LE.D1)GOTO 2
      D0=D1
      K0=I
2     CONTINUE
      RETURN
      END
```

3.3.5 例　　题

1）已知一个四顶点连通图（见图3.19），根据给出的该图各顶点间的距离矩阵与各顶点权值，求该图的中心与加权中心。

$$
D(I,J)=\begin{matrix} 0.0 & 1.0 & 3.0 & 4.0 \\ 1.0 & 0.0 & 2.0 & 90.0 \\ 3.0 & 2.2 & 0.0 & 5.0 \\ 4.0 & 90.0 & 5.0 & 0.0 \end{matrix}
$$

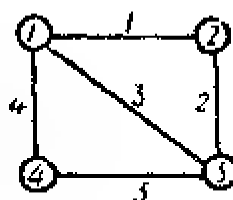


图 3.19

$P(J)=(1.0\ 2.0\ 3.0\ 4.0)$

2）计算程序

```
      REAL D(30,30),P(30)
      WRITE(7,1)
      READ(5,2)N
1     FORMAT(1X,'N=')
2     FORMAT(I3)
      CALL ZZM(N,D,P)
      STOP
      END
      SUBROUTINE ZZM(N,D,P)
      REAL D(N,N),P(N)
```

```
      WRITE(7,2)
2     FORMAT(/,1X,'D(I,J)=')
      DO 4 I=1,N
4     READ(5,9)(D(I,J),J=1,N)
9     FORMAT(30F12.4)
      WRITE(7,12)
12    FORMAT(1X,'P(I)=',/)
      READ(5,9)(P(I),I=1,N)
      WRITE(6,2)
      DO 5 I=1,N
5     WRITE(6,3)(D(I,J),J=1,N)
3     FORMAT(1X,30(F6.2,1X))
      WRITE(6,10) (P(I),I=1,N)
10    FORMAT(/,1X,'P(I)=',/,30(1X,F6.2))
      CALL MINDL0(N,D)
      CALL MIND1(N,D,D1,K0)
      WRITE(6,8)K0,D1
8     FORMAT(1X,'K0=',I2,2X,'D1=',F7.2)
      CALL MIND2(N,D,P,D2,K1)
      WRITE(6,11)K1,D2
11    FORMAT(1X,'K1=',I2,2X,'D2=',F7.2)
      RETURN
      END
```

本程序调用了计算连通图各顶点之间的最短距离矩阵的子程序MINDL0。关于这个子程序的功能，以及它的内容，请参看3.1节的说明。

3）计算结果

K0=1 D1=4.0

K1=1 D2=27.0

由上述计算结果可知，该图的中心是1号顶点，离它自

身最远的顶点是4号顶点，这两个顶点之间的最短距离是4。该图的加权中心也是1号顶点，其余三个顶点至该点的加权距离之和为27。

3.4 图的一般中心的算法

3.4.1 功　能

连通图由N个顶点与M条弧组成。在选址问题中，前面已讨论了服务点与服务对象都在这N个顶点上的情况，称之为寻找图的中心问题。如果服务点只允许取在各顶点上，而服务对象包括各顶点与各弧上的点，则可以在这N个顶点当中选定一个顶点作为图的一般中心，其条件是该点离它本身的最远服务对象（包括顶点与各弧上的点）的距离达到极小值。

将图的中心与图的一般中心进行比较。可以看出，图的一般中心选址方法使服务对象的范围更加广泛了。所以，图的一般中心的选址方法也是一种很有实用价值的算法。

3.4.2 方法概述

设无向连通图的顶点集为$V=\{v_1, v_2, \cdots, v_N\}$，弧集合为$E=\{e_1, e_2, \cdots, e_M\}$。将$e_i$的弧长记作$l(e_i)$。

为了求该图的一般中心，我们采用化整为零的办法。首先求出各顶点之间的最短距离矩阵，然后求出每一个顶点v_i至所有弧e_j上最远点的距离。将这些距离记作d'_{ij}，$i=1, 2, \cdots, N$；$j=1, 2, \cdots, M$。最后对这些距离d'_{ij}进行比较，确定最佳点t_0。具体步骤如下：

步骤1　利用子程序MINDL0(N,D)，首先求出各顶点

的最短距离矩阵D(d_{ik})，$i,k=1,2,\cdots,N$。

步骤2 利用公式$d'_{ij}=0.5*(d_{ik_1}+l(e_j)+d_{ik_2})=0.5*(d_{ik_1}+a_{k_1k_2}+d_{ik_2})$，求出各顶点至各弧上最远点的距离。上式中$i=1,2,\cdots,N$；$j=1,2,\cdots,M$；$k_1$与$k_2$是弧$e_j$的两个顶点的序号，$a_{k_1k_2}$为弧$e_j$的长度。

步骤3 利用公式$d_0=\min\limits_{1\leqslant i\leqslant N}\{\max\limits_{1\leqslant j\leqslant M}\{d'_{ij}\}\}$，求由顶点至各弧最远点的距离达到极小的顶点，记该顶点的序号为i_0。第i_0号顶点是该图的一般中心，计算结束。

3.4.3 子程序参数说明

子程序名称 MYNCN (N,M,AMAX0,D,K0,D0,D2,E)

N：图的顶点数目，整型变量。

M：图上弧的数目，整型变量。

AMAX0：弧长上界，输入数据，取一个比图上全部弧长之和大的正实数。实型变量。

D(N, N)：图上各顶点间的原始距离矩阵的输入数据，实型数组。当第i顶点与第j顶点不邻接，取$D(i,j)=$AMAX0。

K0：图的一般中心顶点顺序号，输出结果，整型变量。

D0：图的一般中心顶点与离它本身最远点的距离，输出结果，实型变量。

D2(N,N)：工作单元，实型数组。

E(M,3)：工作单元，实型数组。

3.4.4 子程序

```
      SUBROUTINE MYNCN(N,M,AMAX0, D, K0, D0,
     # D2, E)
      REAL D(N,N),D2(N,N),E(M,3)
      D0=AMAX0
      DO 1 I=1,N
      DO 1 J=1,N
1     D2(I,J)=D(I,J)
      CALL MINDL0(N,D2)
      K=0
      N1=N-1
      DO 4 I=1,N1
      I1=I+1
      DO 4 J=I1,N
      IF (D(I,J).GE.AMAX0)  GOTO 4
      K=K+1
      E(K,1)=I
      E(K,2)=J
      E(K,3)=D(I,J)
4     CONTINUE
      DO 6 I=1,N
      DM=0.0
      DO 5 K=1,M
      I1=E(K,1)
      I2=E(K,2)
      D1=0.5*(D2(I,I1)+E(K,3)+D2(I,I2))
5     DM=AMAX1(DM,D1)
      IF (D0.LE.DM) GOTO 6
```

```
      D0= DM
      K0=I
6     CONTINUE
      RETURN
      END
```

3.4.5 例　题

1）求以下两图的一般中心。

（1）求五个顶点连通图（见图3.20）的一般中心。

N=5, AMAX0=99.0

$$D(I,J)=\begin{matrix} 0.0 & 3.0 & 2.0 & 99.0 & 99.0 \\ 3.0 & 0.0 & 4.0 & 2.0 & 99.0 \\ 2.0 & 4.0 & 0.0 & 4.0 & 99.0 \\ 99.0 & 2.0 & 4.0 & 0.0 & 1.0 \\ 99.0 & 99.0 & 99.0 & 1.0 & 0.0 \end{matrix}$$

（2）某乡的七个村庄的交通网如图3.21所示。拟在其中一个村内建一个自行车修理部。从所有在道路上行驶的自行车离修理部的距离最近来考虑，这个修理部应当选址建在哪个村庄？

用求图的一般中心的算法来解决这一问题。

N=7, AMAX0=90.0

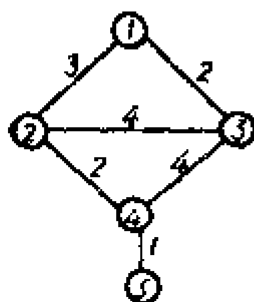


图 3.20

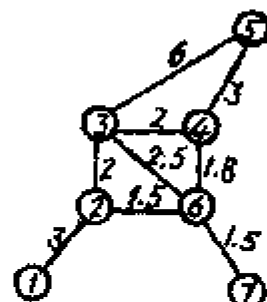


图 3.21

$D(I, J) =$

$$\begin{vmatrix} 0.0 & 3.0 & 90.0 & 90.0 & 90.0 & 90.0 & 90.0 \\ 3.0 & 0.0 & 2.0 & 90.0 & 90.0 & 1.5 & 90.0 \\ 90.0 & 2.0 & 0.0 & 2.0 & 6.0 & 2.5 & 90.0 \\ 90.0 & 90.0 & 2.0 & 0.0 & 3.0 & 1.8 & 90.0 \\ 90.0 & 90.0 & 6.0 & 3.0 & 0.0 & 90.0 & 90.0 \\ 90.0 & 1.5 & 2.5 & 1.8 & 90.0 & 0.0 & 1.5 \\ 90.0 & 90.0 & 90.0 & 90.0 & 90.0 & 1.5 & 0.0 \end{vmatrix}$$

2）计算程序

```
      REAL D(20,20)
      READ(5,*)N,AMAX0
      CALL ZYC(N,D,AMAX0)
      STOP
      END
C
      SUBROUTINE ZYC(N,D,AMAX0)
      REAL D(N,N),D2(20,20),E(30,3)
      READ(5,*)D
      DO 10 I=1,N
      WRITE(6,11)(D(I,J),J=1,N)
11    FORMAT(1X,20F8.2)
10    CONTINUE
      M=0
      DO 1 I=2,N
      II=I-1
      DO 1 J=1,II
      IF(D(I,J).LT.AMAX0)  M=  M+1
1     CONTINUE
      CALL MYNCN(N,M,AMAX0,D,K0,D0,D2,E)
```

```
      WRITE(6,2)K0,D0
2     FORMAT(1X,'K0=',I3,3X,'D0=',F12.4)
      RETURN
      END
```

本程序调用了计算连通图各顶点之间的最短距离矩阵的子程序MINDL0。关于这个子程序的功能以及它的内容，请参看3.1段的说明。

3）计算结果

（1）K0=2

D0=5.0

第2号顶点是该图的一般中心，离一般中心最远的点有两个：一个是顶点1与顶点3之间弧的中点，另一个是顶点3与顶点4之间的弧上距3号顶点距离为1的点。这两个点至2号顶点的距离都是5.0。

（2）K0=3

D0=5.50

图的一般中心在第3号顶点上。图上所有弧中离3号顶点最远的点只有一个，它在3号顶点至5号顶点这段弧的距5号顶点距离0.5的位置。由3号顶点到达此点的距离，无论经过4号顶点与5号顶点这条路，或者沿3号顶点至5号顶点的弧直接到达，距离都是5.50。由图3.21也可以看到，由3号顶点到图上其它任何点的距离都小于5.50。

3.5 求连通图的绝对中心的算法

3.5.1 功　能

图的绝对中心这一算法，常用于服务点的选址问题上。

它是图论当中一个十分有用的算法。所谓求一个连通图的绝对中心的问题，要求服务对象都在该图的各顶点上；而服务点既可以在图的顶点上选，也可以在该图的所有弧上的任意点上选址。只要选到的点满足这样的条件，由这个点到离它自身最远的服务对象所在顶点的最短路径达到极小值，就找到了图的绝对中心。由此可见，图的绝对中心比前面介绍的图的中心和一般中心选址范围更广。

3.5.2 方法概述

已知一个由 n 个顶点 m 条弧构成的无向连通图。首先对顶点进行编号，并依这个顺序排出图上各顶点之间的原始距离矩阵。记 $D(i, j)$ 为第 i 号顶点至第 j 号顶点的弧长。若第 i_0 号与第 j_0 号顶点之间没有弧直接连结，则记作 $D(i_0, j_0) = \infty$。在实际计算时，则以一个大于该图所有弧长之和的实数来代替 ∞。在例题当中，我们暂时以90.0来代替 ∞。使用者可以根据自己数据的要求来改变这个数字的大小。

以下给出求图的绝对中心的计算流程，具体步骤如下：

步骤1 将原始距离矩阵 $D(N, N)$ 中各弧的长度记入矩阵 $GL(m,3)$ 当中。这里的 m 为该图含弧的数目。

步骤2 用 Dijkstra 方法，调用求无向图各顶点间最短距离子程序MINDL0(N, D)，求出各顶点间的最短距离矩阵，其结果仍然存放在 $D(N, N)$ 当中。

步骤3 按照 $GL(M, 3)$ 中各弧的顺序，分别求出 M 条弧的局部绝对中心。第 k 条弧（$k=1,2,\cdots,M$）的局部中心的求解步骤如下：

① 对第 k 条弧，求出它相对图上 N 个顶点的坐标值：

$E(i)=D(i,GL(k,1))$　　　　　　　　称为上升距；

$C(i)=D(i,GL(k,2))+GL(k,3)$，　称为下降距；

$F(i)=0.5*(C(i)-E(i))$，　　　　　称为分界点。

$$i=1,2,\cdots,N。$$

② 对N个上升距求极大值$U=\max\limits_{1\leqslant i\leqslant N}\{E(i)\}$。

③ 同时对$E(i)$，$C(i)$，$F(i)(i=1,2,\cdots,N)$按其值的大小重新排列它们的顺序。排序遵从以下原则：当$E(i)>E(j)$时，$E(i)$，$C(i)$，$F(i)$分别排在$E(j)$，$C(j)$，$F(j)$的前面。当$E(i)<E(j)$时，将$E(i)$与$E(j)$，$C(i)$与$C(j)$，$F(i)$与$F(j)$对换位置。当$E(i)=E(j)$时，比较$C(i)$与$C(j)$的大小。当$C(i)>C(j)$时，仍然是$E(i)$，$C(i)$，$F(i)$排在前面。否则，按前述方法将$E(i)$与$E(j)$，$C(i)$与$C(j)$，$F(i)$与$F(j)$对换位置。以上排序的目的是为了减少求局部中心的计算工作量。

④ 令$i=1$。

⑤ 当$i>N$时，转向下一步骤⑥；否则，顺序往下执行。计算不同弧的上升边与下降边的交点：

$$X=0.5*(C(i)-E(j))\quad j=i+1,\cdots,N。$$

当$F(i)<X<F(j)$并且有$X+E(i)<U$时，使$U=E(i)+X$，$V=X$，$i=j$。返回⑤的开头去。

⑥ 若下降距的几何高度$C(i)-GL(k,3)<U$时，使$U=C(i)-GL(k,3)$，$V=GL(k,3)$。第k条弧的局部中心计算结束。得到的V为该弧上局部中心到弧的一个顶点的距离。U为局部中心点到离它本身最远顶点的距离。

步骤4 逐次比较各弧的局部中心，使$U_{k_0}=\min\limits_{1\leqslant k\leqslant M}\{U_k\}$，则第$k_0$条弧的局部中心就是该图的绝对中心。计算结束。

根据计算结果，马上可以在原图上找到绝对中心的位置。

最后应当指出，对上升距、下降距的重新排序，使得原先要在几何图形上进行的寻找极小点的比较，变为程序中的不超过N次简单易行的逻辑比较。这是本算法在计算机上实现的一个关键步骤。

3.5.3 子程序参数说明

1）子程序名称　MINZYN(N,M,D,M1,M2,S)

求图的绝对中心子程序名。

N：已知连通图顶点数目，整型变量。

M：该图所含弧的数目，整型变量。

D(N, N)：该图的原始距离矩阵的输入数据，实型数组。若第i与第j顶点不邻接，则在$D(i, j)$与$D(j, i)$处都填入一个比图上所有弧长之和还要大的实数。

M1, M2：绝对中心所在弧的两端顶点的序号，是绝对中心计算结果的一部分。整型变量。

S(3)：实型数组，用以存放绝对中心的输出结果。其中$S(1)$为绝对中心点所在弧的弧长，$S(2)$为所求出的绝对中心点至$M1$号顶点的距离。$S(3)$为绝对中心点与离它自身最远的顶点的距离。

2）子程序名称　MINDL0(N,D)

求连通图中各顶点间最短距离子程序名。其参数说明与子程序清单请查阅本书3.1节。

3.5.4 子程序

```
      SUBROUTINE MINZYN(N,M,D,M1,M2,S)
      REAL D(N,N),S(3),GL(20,3),C(20),
   #  E(20),F(20)
      K=0
      II=N-1
      DO 12 I=1,II
      JJ=I+1
      DO 11 J=JJ,N
      IF(D(I,J).GE.90.0)  GOTO 11
      K=K+1
      GL(K,1)=I
      GL(K,2)=J
      GL(K,3)=D(I,J)
11    CONTINUE
12    CONTINUE
      CALL MINDL0(N,D)
      S(1)=0.0
      S(2)=0.0
      S(3)=1.0E9
      DO 25 K=1,M
      DO 14 I=1,N
      L=GL(K,1)
      E(I)=D(I,L)
      L=GL(K,2)
      C(I)=D(I,L)+GL(K,3)
14    F(I)=(C(I)-E(I))×0.5
      U=0.0
```

```
      V=0.0
      DO 15 I=1,N
15    U=AMAX1(U,E(I))
      NN=N-1
      DO 16 I=1,NN
      II=I+1
      DO 17 J=II,N
      IF(E(I).LT.E(J).OR.E(I).EQ.E(J)
   #  .AND.C(I).LT.C(J))GOTO 18
      GOTO 17
18    P=E(I)
      E(I)=E(J)
      E(J)=P
      P=C(I)
      C(I)=C(J)
      C(J)=P
      P=F(I)
      F(I)=F(J)
      F(J)=P
17    CONTINUE
16    CONTINUE
      I=1
20    II=I+1
      DO 21 J=II,N
      X=(C(I)-E(J))*0.5
      IF(F(I).GE.X.OR.X.GE.F(J))GOTO 21
      IF(U.LE.E(J)+X)GOTO 22
      U=E(J)+X
      V=X
22    I=J
      GOTO 20
```

```
21      CONTINUE
        IF(C(I)-GL(K,3).GE.U)  GOTO 23
        U=C(I)-GL(K,3)
        V=GL(K,3)
23      IF(S(3).LE.U)  GOTO 24
        S(3)=U
        M1=GL(K,1)
        M2=GL(K,2)
        S(1)=GL(K,3)
        S(2)=V
24      CONTINUE
25      CONTINUE
        RETURN
        END
```

3.5.5 例　题

1）求以下三个图的绝对中心。将绝对中心的计算结果标在原图上。

（1）见图3.22。

$N=4$，$M=5$

$$D(I,J)=\begin{pmatrix} 0.00 & 8.00 & 99.00 & 3.00 \\ 8.00 & 0.00 & 2.00 & 4.00 \\ 99.00 & 2.00 & 0.00 & 3.00 \\ 3.00 & 4.00 & 3.00 & 0.00 \end{pmatrix}$$

（2）见图3.23。

$N=5$，$M=6$

$$D(I,J)=\begin{pmatrix} 0.00 & 3.00 & 2.00 & 99.00 & 99.00 \\ 3.00 & 0.00 & 4.00 & 2.00 & 99.00 \\ 2.00 & 4.00 & 0.00 & 4.00 & 99.00 \\ 99.00 & 2.00 & 4.00 & 0.00 & 1.00 \\ 99.00 & 99.00 & 99.00 & 1.00 & 0.00 \end{pmatrix}$$

（3）见图3.24。

$N=5$，$M=5$

$$D(I,J)=\begin{pmatrix} 0.00 & 1.00 & 99.00 & 99.00 & 1.80 \\ 1.00 & 0.00 & 1.20 & 99.00 & 99.00 \\ 99.00 & 1.20 & 0.00 & 1.40 & 99.00 \\ 99.00 & 99.00 & 1.40 & 0.00 & 1.60 \\ 1.80 & 99.00 & 99.00 & 1.60 & 0.00 \end{pmatrix}$$

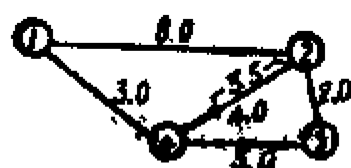


图 3.22

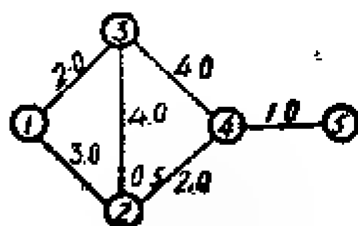


图 3.23

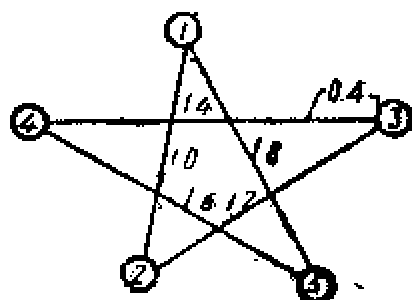


图 3.24

2）计算程序

```
      REAL D(20,20),S(3)
      READ(5,30)N
30    FORMAT(I4)
      CALL ZYY(N,D,S)
      STOP
      END
C
```

```
      SUBROUTINE ZYY(N,D,S)
      REAL D(N,N),S(3)
      M=0
      DO 1 I=2,N
      II=I-1
      READ(5,*)(D(I,J),J=1,II)
      DO 1 J=1,II
      D(J,I)=D(I,J)
      IF(D(I,J).GE.90.0)GOTO 1
      M=M+1
1     CONTINUE
      DO 3 I=1,N
3     D(I,I)=0.0
      CALL MINZYN(N,M,D,M1,M2,S)
      WRITE(6,8)M1,M2,S
8     FORMAT(1X,2I5,3F9.2)
      RETURN
      END
```

在程序中，暂时用90.0作为判断各弧的长度是否为∞的一个上界值。在使用此程序解决读者的实际问题时，可根据自己题目数据的量级，改变这个上界值，以免造成错误。

3）计算结果

（1）

M1=2，M2=4

S(1)=4.00，S(2)=3.50，S(3)=3.50。

（2）

M1=2，M2=3

S(1)=4.00，S(2)=0.50，S(3)=3.50。

（3）

M1=3，M2=4

S(1) = 1.40, S(2) = 0.40, S(3) = 2.60。

3.6 图上两顶点间最短与次短路的计算

3.6.1 功　能

前面介绍了在连通图上求两顶点间的最短通路的算法。许多实际问题不仅需要求最短路，还要找到它的次短路。本节将介绍次短路的求法。如果相同出发点S至相同终点的相等距离的次短路不只一条，本节程序得到的是与最短路公共弧最多的一条次短路。

3.6.2 方法概述

已知无向或有向连通图的各顶点间的距离矩阵$A(n, n)$。对于两顶点间不邻接的情况，矩阵A的相应元素存放一个比弧长上界$M0$更大的数字。为了求该图中由指定出发点S至终点T的次短路，应当先求出S至T的最短路。我们只要使用前面讲过的子程序 DIJK11 进行计算就可以得到这条最短路。将最短路所经过的顶点编号顺序存放在$D0(n)$当中，将路长存放于$D00$。假定最短路包含n_0条有向弧。我们每次删去原距离距阵A当中最短路中的一条有向弧，就得到n_0个与A只有一个元素差别的新距离矩阵。对这n_0个矩阵分别求出一条S至T的最短路。由这n_0条最短路中选最短路长的一条，作为次短路的解。以下，给出计算流程图3.25。

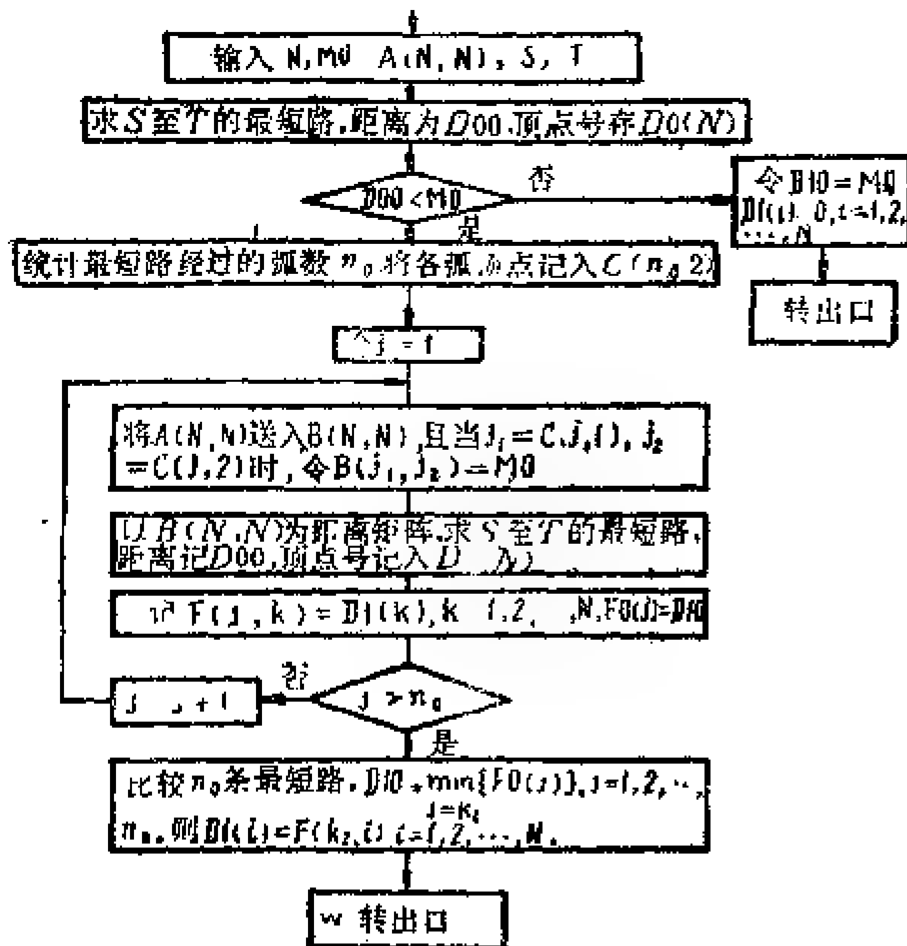


图 3.25

3.6.3 子程序参数说明

子程序名称　DIJK12(N,A,B,D0,D00,D1,D10,S,T,M0,N1,N2)

N: 图的顶点数目，整型变量。

A(N,N): 图上各顶点间的原始距离矩阵，整型数组。详见子程序DIJK11参数说明。

B(N,N): 整型数组，中间工作单元。

D0(N): 存放由 S 至 T 点的最短路顺序经过的顶点序号，计算结果，整型数组。当最短路顶点数小于 N 时，空

余单元内补零。

D00：由 S 至 T 的最短路的长度，整型变量。

D1(N)：存放由 S 至 T 点的次短路顺序经过的顶点序号。用法与D0(N)相同，整型数组。

D10：由 S 至 T 的次短路的长度，整型变量。

S：最短路与次短路的出发点，整型变量。

T：最短路与次短路的终点，整型变量。

M0：弧长上界的输入数据，置一个大于所有弧长之和的整数，整型变量。

N1：由 S 至 T 最短通路经过的顶点数目，整型变量。

N2：由 S 至 T 次短路经过的顶点数目，整型变量。

3.6.4 子 程 序

```
      SUBROUTINE DIJK12(N,A,B,D0,D00,D1,D10,S,
   #  T,M0,N1,N2)
      INTEGER S,T,D00,D10,A(N,N),B(N,N),D0(N),
   #  D1(N),F(20,20),F0(20),W,C(20,2)
      DO 1 I=1,N
      D1(I)=0
      C(I,1)=0
      C(I,2)=0
      DO 2 J=1,N
2     F(I,J)=0
1     F0(I)=0
      D10=0
      CALL DIJK11(N,A,D0,D00,S,T,M0,N1)
      IF(D00.EQ.M0)D10=M0
      IF(D10.EQ.M0)N2=0
```

```
      IF(D10.EQ.M0)GOTO 20
      N0 = 0
      DO 3 I = 2,N
      IF(D0(I).EQ.0) GOTO 3
      NO = N0 + 1
      C(N0,1) = D0( . - 1)
      C(N0,2) = D0(I)
3     CONTINUE
      DO 4 I = 1,N
      DO 4 J = 1,N
4     B(I,J) = A(  ,J)
      DO 5 J = 1,N0
      J1 = C(J,1)
      J2 = C(J,2)
      B(J1,J2) = M0
      CALL DIJK11(N,B,D1,D10,S,J2,M0,N2)
      B(J1,J2) = A(J1,J2)
      F0(J) - MIN0(M0,D10)
      DO 6 K = 1,N2
6     F(J,K) = D1(K)
      J1 = J + 1
      IF(J1.GT.N0) GOTO 8
      DO 7 K = J1,N0
      N2 - N2 + 1
      K1 = D0(K)
      K2 = D0(K + 1)
      F(J,N2) = F(J,N2) + K2
7     F0(J) = F0(J) + A(K1,K2)
5     CONTINUE
8     W = M0
      DO 9 J = 1,N0
```

```
      IF(W.LE.F0(J))  GOTO 9
      W=F0(J)
      K2=J
9     CONTINUE
      N2=0
      DO 10 I=1,N
      IF (F(K2,I).NE.0) N2=N2+1
10    D1(I)=F(K2,I)
      D10=F0(K2)
      GOTO 20
12    DO 14 I=1,N
14    D1(I)=0
      D10=M0
      N2=0
20    CONTINUE
      RETURN
      END
```

3.6.5 例　　题

1）求以下两图各顶点之间的最短路和次短路。

（1）见图3.26。

$N=4$，$M0=800$

$$A(I,J)=\begin{vmatrix} 0 & 6 & 900 & 4 \\ 6 & 0 & 2 & 3 \\ 900 & 2 & 0 & 5 \\ 4 & 3 & 5 & 0 \end{vmatrix}$$

求各顶点之间的最短路和次短路，并对$S=1$，$T=3$；

$S=2$，$T=3$；$S=3$，$T=4$；$S=4$，$T=1$的结果作图。

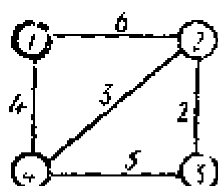


图 3.26

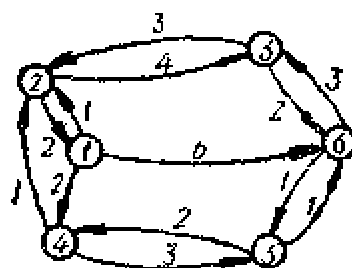


图 3.27

（2）见图3.27。$N=6$，$M0=800$

$$A(I,J)=\begin{pmatrix} 0 & 1 & 900 & 2 & 900 & 6 \\ 2 & \blacksquare & 4 & 900 & 900 & 900 \\ 900 & 3 & 0 & 900 & 900 & 2 \\ 900 & 1 & 900 & 0 & 3 & 900 \\ 900 & 900 & 900 & 2 & 0 & 1 \\ 900 & 900 & 3 & 900 & 1 & 0 \end{pmatrix}$$

求各顶点之间的最短路和次短路，并对$S=1$，$T=3$，$S=2$，$T=4$；$S=3$，$T=1$；$S=4$，$T=2$；$S=5$，$T=6$；$S=6$，$T=5$的结果作图。

2）计算程序

```
     INTEGER S,T,D00,A(20,20),D0(20),D1(20),
#    B(20,20)
     WRITE(7,20)
     READ(5,21) N
     WRITE(7,22)
     READ(5,21) M0
     CALL DIJ(N,M0,A,B,D0,D1)
```

```
20      FORMAT(1X,2HN=)
21      FORMAT(I4)
22      FORMAT(1X,3HM0=)
        STOP
        END
        SUBROUTINE DIJ(N,M0,A,B,D0,D1)
        INTEGER S,T,D00,D10,D1(N),D0(N),A(N,N),
     #  B(N,N)
        WRITE(7,20)
20      FORMAT(1X,'A(I,J)=',1)
        DO 21 I=1,N
21      READ(5,*)(A(I,J),J=1,N)
        WRITE(6,19)N,M0
19      FORMAT(1X,'N=',I3,M0=',I4/1)
        WRITE(6,20)
        DO 22 I=1,N
22      WRITE(6,23)(A(I,J),J=1,N)
23      FORMAT(1X,20 I4)
        WRITE(6,24)
24      FORMAT(/)
        DO 26 S=1,N
        DO 26 T=1,N
        IF (S.EQ.T) GOTO 26
        CALL DIJK12(N,A,B,D0,D00,D1,D10,S,
     #  T,M0,N1,N2)
        WRITE(6,31)S,T,D00,(D0(I),I=-1,N1)
31      FORMAT(1X,2HS=,I2,1X,2HT=,I2,1X,
     #  3HD1=,I4,3X,20 I3)
        WRITE(6,44)D10,(D1(I),I=1,N2)
44      FORMAT(1X,10X,2HD2=,I4,3X,20 I3)
26      CONTINUE
```

```
RETURN
END
```

3）计算结果

（1）分别见图3.28～图3.31。

```
S=1   T=2   D1=6    1 2
            D2=7    1 4 2
S=1   T=3   D1=8    1 2 3
            D2=9    1 4 2 3
S=1   T=4   D1=4    1 4
            D2=9    1 2 4
S=2   T=1   D1=6    2 1
            D2=7    2 4 1
S=2   T=3   D1=2    2 3
            D2=8    2 4 3
S=2   T=4   D1=3    2 4
            D2=7    2 3 4
S=3   T=1   D1=8    3 2 1
            D2=9    3 4 1
S=3   T=2   D1=2    3 2
            D2=8    3 4 2
S=3   T=4   D1=5    3 4
            D2=5    3 2 4
S=4   T=1   D1=4    4 1
            D2=9    4 2 1
S=4   T=2   D1=3    4 2
            D2=7    4 3 2
S=4   T=3   D1=5    4 3
            D2=5    4 2 3
```

（2）分别见图3.32～图3.37。

```
S=1   T=2   D1=1    1 2
```

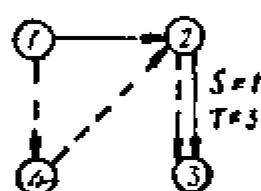


图 3.28

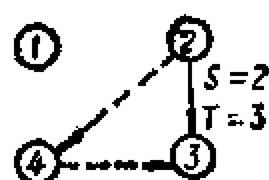


图 3.29

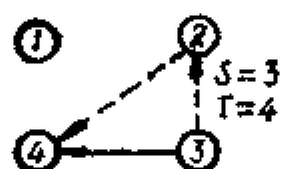


图 3.30

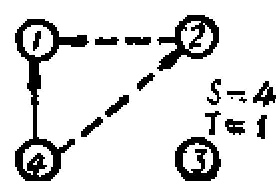


图 3.31

		D2 = 3	1 4 2
S = 1	T = 3	D1 = 5	1 2 3
		D2 = 7	1 4 2 3
S = 1	T = 4	D1 = 2	1 4
		D2 = 9	1 6 5 4
S = 1	T = 5	D1 = 5	1 4 5
		D2 = 7	1 6 5
S = 1	T = 6	D1 = 6	1 6
		D2 = 6	1 4 5 6
S = 2	T = 1	D1 = 2	2 1
		D2 = 800	0
S = 2	T = 3	D1 = 4	2 3
		D2 = 11	2 1 6 3
S = 2	T = 4	D1 = 4	2 1 4
		D2 = 9	2 3 6 5 4
S = 2	T = 5	D1 = 7	2 1 4 5
		D2 = 7	2 3 6 5
S = 2	T = 6	D1 = 6	2 3 6
		D2 = 8	2 1 6

S = 3 T = 1 D1 = 5 3 2 1
D2 = 8 3 6 5 4 2 1
S = 3 T = 2 D1 = 3 3 2
D2 = 6 3 6 5 4 2
S = 3 T = 4 D1 = 5 3 6 5 4
D2 = 7 3 2 1 4
S = 3 T = 5 D1 = 3 3 6 5
D2 = 10 3 2 1 4 5
S = 3 T = 6 D1 = 2 3 6
D2 = 11 3 2 1 6
S = 4 T = 1 D1 = 3 4 2 1
D2 = 12 4 5 6 3 2 1
S = 4 T = 2 D1 = 1 4 2
D2 = 10 4 5 6 3 2
S = 4 T = 3 D1 = 5 4 2 3
D2 = 7 4 5 6 3
S = 4 T = 5 D1 = 3 4 5
D2 = 8 4 2 3 6 5
S = 4 T = 6 D1 = 4 4 5 6
D2 = 7 4 2 3 6
S = 5 T = 1 D1 = 5 5 4 2 1
D2 = 9 5 6 3 2 1
S = 5 T = 2 D1 = 3 5 4 2
D2 = 7 5 6 3 2
S = 5 T = 3 D1 = 4 5 6 3
D2 = 7 5 4 2 3
S = 5 T = 4 D1 = 2 5 4
D2 = 11 5 6 3 2 1 4
S = 5 T = 6 D1 = 1 5 6
D2 = 9 5 4 2 3 6
S = 6 T = 1 D1 = 6 6 5 4 2 1

		D2 = 8	6 8 2 1
S = 6	T = 2	D1 = 4	6 5 4 2
		D2 = 6	6 8 2
S = 6	T = 3	D1 = 3	6 8
		D2 = 8	6 5 4 2 3
S = 6	T = 4	D1 = 8	6 5 4
		D2 = 10	6 3 2 1 4
S = 6	T = 5	D1 = 1	6 5
		D2－13	6 3 2 1 4 5

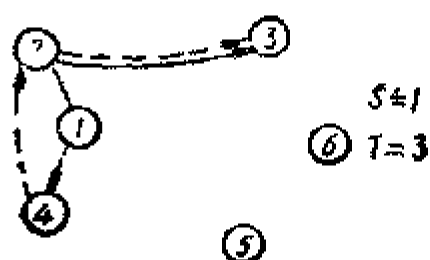


图 3.32

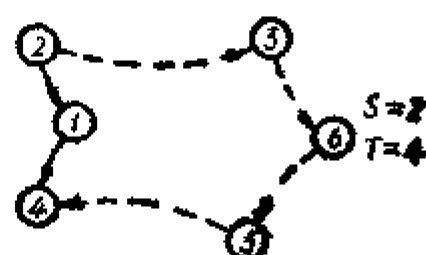


图 3.33

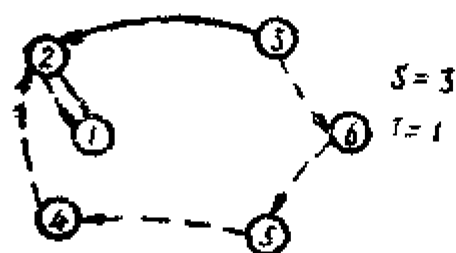


图 3.34

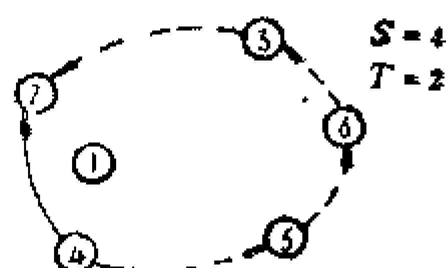


图 3.35

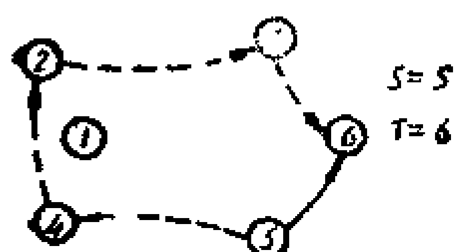


图 3.36

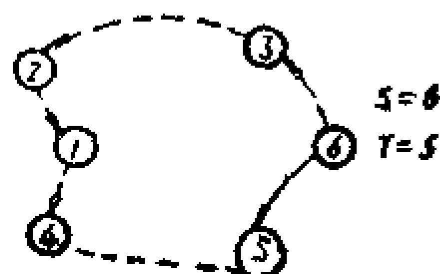


图 3.37

在以上例题中，各图当中的实线标出S至T的最短路线与方向，虚线标出次短路线与方向。

3.7 最大容量路的算法

3.7.1 功　能

最大容量路属于网络最短路的另一变种形式。在交通运输网络中，每一段弧都可以给予一定数值的通过能力，我们称这个通过能力的值为该弧的容量。网络中，任意两个顶点间如果有一条通路，该通路的容量大小应当等于该通路上所有弧段的容量值当中最小的那个容量值。寻找一条由出发点S至终点T的最大容量路的问题，也就是找一条由S至T的一条通路，使这一通路上容量值最小的那一段弧的容量值在全部由S至T的通路当中达到极大值。最大容量路这一算法，在主要考虑运输量的交通运输问题上是十分有用的。

3.7.2 方法概述

假定我们已经有了求网络图中由指定出发点S至终点T的最短路的算法。只要将输入的原始距离矩阵的数据作适当的修改，用各弧的容量代替各弧的距离值，并对各弧容量作如下规定：

$$A(i,j)=\begin{cases} M, & \text{当 } i=j \text{ 时；} \\ a_{ij}, & \text{当第} i \text{点至第} j \text{点的弧段容量值为 } a_{ij}\text{；} \\ 0, & \text{当第 } i \text{ 点至第 } j \text{ 点不相邻接。} \end{cases}$$

$$i,j=1,2\cdots,N。$$

这里的M是容量上界。然后利用前一节计算出发点S至终点

T最短路的子程序 DIJK11。对该子程序先作下面的改动，然后使用它求最大容量路。

1）初态的修改

```
D(I, 2)
D(S, 2)  M
G = 0
```

2）取极值条件的改动

```
V  MIN0(D(J,2),A(J,I))
W=MAX0(U,V)
```

3）判断条件的改动

```
IF(W.LE.0) GOTO 3
IF(U.LT.V) D(I,3)=J
IF(W.LE.G) GOTO 3
IF(I0.EQ.0) GOTO 2
IF(D00.EQ.0) GOTO 12
```

本节提供的子程序就是这样一个经过改动上述内容的子程序。

3.7.3 子程序参数说明

子程序名称　MAXCL(N, A, D0, D00, S, T, M)

N：图的顶点数目，整型变量。

A(N, N)：图上各弧容量值的输入矩阵。当第 i 与第 j 顶点间无弧邻接时，取 $A(i,j)=0$。令 $A(i,i)=M$，$i=1,2,\cdots,N$。整型数组。

D0(N)：最大容量路所经过顶点序号的输出结果，整型数组。如果该容量路经过的顶点数目少于 N 时，该数

组多余单元补零。

D00：最大容量路的允许容量的计算结果，整型变量。

S：求最大容量路的出发点序号，输入数据，整型变量。

T：求最大容量路的终点序号，输入数据，整型变量。

M：容量上界，取一个比网络中所有弧的容量值更大的常数，整型变量。

注意：以上全部参数均使用了整型量，如果使用者需要处理的容量数据为实数，可以将数据放大后，作为整数来处理。

3.7.4 子程序

```
      SUBROUTINE MAXCL(N,A,D0,D00,S,T,M)
      INTEGER S,T,D00,G,W,U,V,D0(N),
     # A(N,N),D(20,3)
      D00=0
      DO 1 I=1,N
      D0(I)=0
      D(I,1)=0
      D(I,2)=0
1     D(I,3)=S
      D(S,1)=1
      D(S,2)=M
      J=S
2     G=0
      I0=0
      DO 3 I=1,N
      IF(D(I,1).NE.0) GOTO 3
      U=D(I,2)
```

```
      V=MIN0(D(J,2),A(J,I))
      W=MAX0(U,V)
      D(I,2)=W
      IF(W.LE.0) GOTO 3
      IF(U.LT.V) D(I,3)=J
      IF(W.LE.G) GOTO 3
      I0=I
      G=W
3     CONTINUE
      IF(I0.EQ.0) GOTO 12
      D(I0,1)= 1
      J=I0
      IF (D(T,1).EQ.0) GOTO 2
      D00=D(T,2)
      D0(1)=T
      J=T
      K=2
      IF(D00.EQ.0) GOTO 12
5     D0(K)=D(J,3)
      J=D(J,3)
      K=K+1
      IF(J.NE.S)GOTO 5
12    CONTINUE
      NN=K-1
      IF(NN/2.EQ.NN/2.)NM=NN/2
      IF(NN/2.NE.NN/2.)NM=(NN-1)/2
      DO 22 K=1,NM
      KK=D0(K)
      JJ=NN-K+1
      D0(K)=D0(JJ)
22    D0(JJ)=KK
```

```
      RETURN
      END
```

3.7.5 例　　题

1）已知五点运输网络各弧段的运输容量(见图3.38)，求低序号顶点至高序号顶点的最大容量路。

$$N=5,\ M=90$$

$$A(I,J)=\begin{bmatrix} 90 & 2 & 0 & 6 & 10 \\ 3 & 90 & 4 & 2 & 0 \\ 5 & 6 & 90 & 3 & 0 \\ 0 & 5 & 5 & 90 & 8 \\ 2 & 0 & 7 & 7 & 90 \end{bmatrix}$$

2）计算程序.

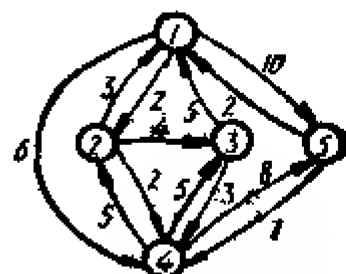


图 3.38

```
      INTEGER A(20,20),D0(20)
      WRITE(7,1)
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)M
1     FORMAT(1X,2HN=)
2     FORMAT(1X,2HM=)
      CALL MAC(N,M,A,D0)
      STOP
      END
      SUBROUTINE MAC(N,M,A,D0)
      INTEGER S,T,D00,D0(N),A(N,N)
      WRITE(7,3)
3     FORMAT(1X,7HA(I,J)=)
      DO 2 I=1,N
```

```
2        READ(5,4)(A(I,J),J=1,N)
4        FORMAT(20I5)
         DO 5 I=1,N
         WRITE(6,1)(A(I,J),J=1,N)
1        FORMAT(1X,20I5)
5        CONTINUE
         DO 7 S=1,N
         DO 7 T=S,N
         IF (S.EQ.T) GOTO 7
         CALL MAXCL(N,A,D0,D00,S,T,M)
         WRITE(6,6)S,T,D00,(D0(I),I=1,N)
6        FORMAT(1X,2HS=,I2,1X,2HT-,I2,1X,
     #   2HD=,I4,3X,20I3)
7        CONTINUE
         RETURN
         END
```

3）计算结果

```
S= 1   T= 2   D= 6      1 5 3 2 0
S= 1   T= 3   D= 7      1 5 3 0 0
S= 1   T= 4   D= 7      1 5 4 0 0
S= 1   T= 5   D=10      1 5 0 0 0
S= 2   T= 3   D= 4      2 3 0 0 0
S= 2   T= 4   D= 4      2 3 1 4 0
S= 2   T= 5   D= 4      2 3 1 5 0
S= 3   T= 4   D= 5      3 1 4 0 0
S= 3   T= 5   D= 5      3 1 5 0 0
S= 4   T= 5   D= 8      4 5 0 0 0
```

3.8 最大可靠路的算法

3.8.1 功　能

在给定的通讯或运输网络中，已知各弧的可靠性概率值，使用本节的程序可以求出由网络中指定的出发点S至终点T的一条总的可靠率达到最大值的通路。我们称这条通路为由S点至T点的最大可靠路。

3.8.2 方法概述

最大可靠路的算法是最短通路算法的一个变种。你只要了解前面介绍过的求网络中S至T的最短通路的算法，将网络各顶点间的输入数据由原始距离改为可靠率值，并将这些概率值作适当的变换，按照求最短路的步骤就可以解决求最大可靠路的问题。

已知网络G上各顶点间相通弧段的完好概率为$0 \leqslant p_{ij} \leqslant 1$。设一条由顶点$S$至顶点$T$的通路所经过的顶点序号依次为$\{S, S_1, S_2, \cdots, S_k, T\}$，则这条通路所经过的各弧段的完好概率分别是$p_{SS_1}, p_{S_1S_2}, p_{S_2S_3}, \cdots, p_{S_kT}$。这条通路的总完好概率应当是它经过的所有弧段完好概率的乘积：

$$\tilde{p}_{ST} = p_{SS_1} \times p_{S_1S_2} \times p_{S_2S_3} \times \cdots \times p_{S_kT}$$

我们的目标成为寻找一条使$\tilde{p}_{ST}$取极大值的通路的问题了。为此，作变换：

$$a_{ij} = \begin{cases} -\ln p_{ij}, & \text{当} \quad 1 > p_{ij} > 0 \text{ 时；} \\ M, & \text{当} \quad p_{ij} = M > 1 \text{ 时；} \\ 0, & \text{当} \quad p_{ij} = 1 \text{ 或 } i = j \text{ 时。} \end{cases}$$

$i, j = 1, 2, \cdots, N$。

应当指出，当网络中某两点不邻接时，认为完好概率为零，此时在输入的完好概率矩阵的相应元素位置填写的数据是一个大于1的指定数值M，而不填零($p_{ij}=M>1$)。

在作过上述变换之后，就可以用Dijkstra标号法对矩阵$A=(a_{ij})$去求S至T的最短路了。求出的这条最短路就是原网络的最大可靠路。其可靠概率为：

$$\tilde{p}_{ST}=\exp(-d_{ST})$$

其中d_{ST}为矩阵A中S至T的“最短路长”的计算结果。

3.8.3 子程序参数说明

子程序名称 MAXPL(N,P,D0,D00,S,T,CM)

N：图的顶点数目，整型变量。

P(N,N)：图上各弧完好概率的输入矩阵，实型数组。当第i顶点与第j顶点不邻接时，取$p(i,j)=CM>1$。$p(i,i)=1$，$i=1,2,\cdots,N$。

D0(N)：最大可靠路所经过的全部顶点序号存放于此数组，输出数据，整型数组。当这条通路经过的顶点数目少于N时，多余的单元补零。

D00：最大可靠路的总完好概率值，计算结果，实型变量。

S：指定的最大可靠路的出发点，整型变量。

T：指定的最大可靠路的终点，整型变量。

CM：凡是图上两个顶点间不邻接时，完好概率矩阵的相应分量处填写的确定数值。要求$CM>1$，实型变量。

3.8.4 子程序

```
      SUBROUTINE MAXPL(N,P,D0,D00,S,T,CM)
      INTEGER S,T,D0(N)
      REAL P(N,N), A(20,20), D(20,3)
      DO 8 I=1,N
      DO 8 J=1,N
      A(I,J)=P(I,J)
      IF(P(I,J).LT.CM)A(I,J)=-ALOG(P(I,J))
8     CONTINUE
      D00=0.0
      DO 1 I=1,N
      D0(I)=0
      D(I,1)=0
      D(I,2)=CM
1     D(I,3)=S
      D(S,1)=1
      D(S,2)=0
      J=S
2     G=CM
      I0=0
      DO 3 I=1,N
      IF(D(I,1).GE. 0.0001)GOTO 3
      U=D(I,2)
      V=D(J,2)+A(J,I)
      W=AMIN1(U,V)
      D(I,2)=W
      IF (W.GE.CM) GOTO 3
      IF (U.GT.V) D(I,3)=J
```

```
      IF (W.GE.G) GOTO 3
      I0=I
      G=W
3     CONTINUE
      IF (I0.EQ.0) GOTO 12
      D (I0,1)=1
      J=I0
      IF (D(T,1).LT. 0.0001) GOTO 2
      IF (D(T,2)-CM) 6,4,4
4     D00=CM
      GOTO 12
6     D00=EXP (-D(T,2))
      D0(1)=T
      J=T
      K=2
5     D0(K)=D(J,3)
      J=D0(K)
      K=K+1
      IF (J.NE.S) GOTO 5
12    CONTINUE
      NN=K-1
      IF (NN/2.EQ.NN/2.) NM=NN/2
      IF (NN/2.NE.NN/2.) NM=(NN-1)/2
      DO 22 K=1, NM
      KK=D0(K)
      JJ=NN-K+1
      D0(K)=D0(JJ)
22    D0(JJ)=KK
      RETURN
      END
```

3.8.5 例　题

1）已知包含八个顶点的无向连通运输网（见图3.39），各弧段的可靠概率列于矩阵P当中，指定$CM=900.0$，求各顶点间的最大可靠路。

$N=8$，$CM=900.0$

$$P(I,J)=\begin{pmatrix}
1.00 & 0.90 & 0.60 & 0.30 & 900.00 & 900.00 & 900.00 & 900.00\\
0.90 & 1.00 & 0.70 & 900.00 & 900.00 & 900.00 & 0.10 & 900.00\\
0.60 & 0.70 & 1.00 & 0.50 & 0.80 & 900.00 & 900.00 & 900.00\\
0.30 & 900.00 & 0.50 & 1.00 & 900.00 & 0.40 & 900.00 & 900.00\\
900.00 & 900.00 & 0.80 & 900.00 & 1.00 & 0.70 & 0.50 & 0.15\\
900.00 & 900.00 & 900.00 & 0.40 & 0.70 & 1.00 & 900.00 & 0.60\\
900.00 & 0.10 & 900.00 & 900.00 & 0.50 & 900.00 & 1.00 & 0.20\\
900.00 & 900.00 & 900.00 & 900.00 & 0.15 & 0.60 & 0.20 & 1.00
\end{pmatrix}$$

2）计算程序

```
      INTEGER D0(20)
      REAL P(20,20)
      WRITE(7,1)
1     FORMAT(1X,2HN=)
      READ(5,2) N
2     FORMAT(I4)
```

```
      WRITE(7,3)
3     FORMAT(1X,3HCM=)
      READ(5,4) CM
4     FORMAT(F9.1)
      WRITE(6,5) N,CM
5     FORMAT(1X,'N=',I3,'    CM=',F9.1)
      CALL MAXP(N,CM,P,D0)
      STOP
      END
C
      SUBROUTINE MAXP(N,CM,P,D0)
      INTEGER S,T,D0(N)
      REAL P(N,N)
      WRITE(7,4)
4     FORMAT(1X,7HP(I,J)=)
      DO 3 I=1,N
3     READ(5,5) (P(I,J),J=1,N)
5     FORMAT(20F11.4)
      WRITE(6,4)
      DO 1 I=1,N
      WRITE(6,2) (P(I,J),J=1,N)
2     FORMAT(1X,20F8.2)
1     CONTINUE
      DO 7 S=1,N
      DO 7 T=S,N
      IF (S.EQ.T) GOTO 7
      CALL MAXPL(N,P,D0,D00,S,T,CM)
      WRITE(6,6) S,T,D00,(D0(I),I=1,N)
6     FORMAT(1X,2HS=,I2,1X,2HT=,I2,1X,
    # 2HP=,F9.3,4X,20I3)
7     CONTINUE
```

```
      RETURN
      END
```

3）计算结果

```
S=1  T=2  P=0.900   1 2
S=1  T=3  P=0.630   1 2 3
S=1  T=4  P=0.315   1 2 3 4
S=1  T=5  P=0.504   1 2 3 5
S=1  T=6  P=0.353   1 2 3 5 6
S=1  T=7  P=0.252   1 2 3 5 7
S=1  T=8  P=0.212   1 2 3 5 6 8
S=2  T=3  P=0.700   2 3
S=2  T=4  P=0.350   2 3 4
S=2  T=5  P=0.560   2 3 5
S=2  T=6  P=0.392   2 3 5 6
S=2  T=7  P=0.280   2 3 5 7
S=2  T=8  P=0.235   2 3 5 6 8
S=3  T=4  P=0.500   3 4
S=3  T=5  P=0.800   3 5
S=3  T=6  P=0.560   3 5 6
S=3  T=7  P=0.400   3 5 7
S=3  T=8  P=0.336   3 5 6 [illegible]
S=4  T=5  P=0.400   4 3 5
S=4  T=6  P=0.400   4 6
S=4  T=7  P=0.200   4 3 5 7
S=4  T=8  P=0.240   4 6 8
S=5  T=6  P=0.700   5 6
S=5  T=7  P=0.500   5 7
S=5  T=8  P=0.420   5 6 8
S=6  T=7  P=0.350   6 5 7
S=6  T=8  P=0.600   6 8
S=7  T=8  P=0.210   7 5 6 8
```

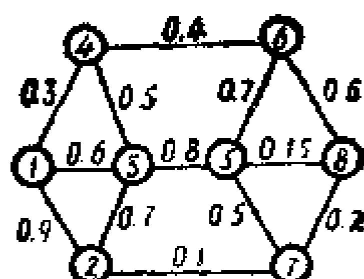


图 3.39

3.9 最大期望容量路的算法

3.9.1 功　　能

前面已介绍了求网络中指定出发点S至终点T的最大可靠路与最大容量路的算法。如果实际问题中需要既考虑到通路的完好概率、又考虑到全通路的运输容量，这就是本节的最大期望容量路的算法。

求最大期望容量路的判断准则是：这条通路前最大通过能力（容量）与该通路的完好概率的乘积达到极大。期望容量的计算公式可以写成下式：

$$f(L)=\left(\prod_{(i,j)\in L} P_{ij}\right)\times \min_{(i,j)\in L}\left\{C_{ij}\right\}$$

上式当中，L代表由出发点S至终点T的一条通路；P_{ij}为这条通路上的一条弧的完好概率；C_{ij}为达条通路上一弧段的容量。最大期望容量路使$f(L)$达到极大。

最大期望容量路的算法在交通运输网络、可靠性通讯网络等方面都是十分有用的。

3.9.2 方法概述

我们采取选取指定发点至终点间的最大可靠路，并根据通路的容量值逐次修改通路的办法来得到期望容量最大的路 L。

已知给定网络的可靠性概率矩阵为 $P(N,N)$，而容量矩阵为 $C(N,N)$。指定的出发点为 S，终点为 T。处理方法归纳为以下 3 步：

步骤 1

① 选取一条由 S 至 T 的最大可靠路，记作 L。如果不存在这样一条通路 L，则转步骤 3。

② 计算路 L 的容量：$C_L=\min\limits_{(i,j)\in L}\{C_{ij}\}$

③ 计算路 L 的期望容量：

$$f(L)=C_L*\prod_{(i,j)\in L}P_{ij}$$

步骤 2

在矩阵 P 与矩阵 C 当中去掉容量小于 C_L 的那些弧。返回步骤 1。

步骤 3

在前面逐次选取的由 S 至 T 的所有通路 L 当中，选取满足 $L_0=\max\{f(L)\}$ 者。L_0 就是最大期望容量路。计算结束。

3.9.3 子程序参数说明

子程序名称

MAXPC(N,CM,P,Q,D,C,D0,D00,S,T)

N：网的顶点数目，整型变量。

CM：完好概率与容量的上界，取一个任何容量值都达不到的大实数，实型变量。必须满足 $CM>1$。

P(N,N)：图上各弧完好概率值矩阵的输入数据，当第 i 与第 j 点不邻接时，取 $P(i,j)=CM>1$。实型数组。

Q(N,N)：中间工作数组，实型数组。

D(N)：中间工作数组，整型数组。

C(N,N)：图上各弧的容量矩阵的输入数据，当第 i 与第 j 点不邻接时，取 $C(i,j)=0$；取 $C(i,i)=CM$，$i=1,2,\cdots,N$。实型数组。

D0(N)：最大期望容量路所经过顶点序号的输出结果，整型数组。当该通路经过的顶点数目少于 N 个时，多余的单元补零。

D00：最大期望容量值的计算结果，实型变量。

S：给定的出发点，整型变量。

T：给定的终点，整型变量。

注：本子程序调用了求最大可靠路子程序，请参看本书的3.8的有关介绍。

3.9.4 子 程 序

```
      SUBROUTINE MAXPC(N,CM,P,Q,D,C,D0,
   #  D00,S,T)
      INTEGER S,T,D0(N),D(N)
      REAL P(N,N),Q(N,N),C(N,N),B(20,20)
      D00=0.0
      DO 2 I=1,N
      D0(I)=0
      DO 1 J=1,N
```

```
      Q(I,J)=P(I,J)
1     B(I,J)=C(I,J)
2     CONTINUE
3     CALL MAXPL(N,Q,D,D10,S,T,CM)
      IF (D10.LT.1E-10) GOTO 12
      G=CM
      DO 4 I=2,N
      IF (D(I).EQ.0) GOTO 4
      K2=D(I-1)
      K1=D(I)
      G=AMIN1(G,B(K1,K2))
4     CONTINUE
      D10=D10*G
      IF(D10.LE.D00) GOTO 6
      D00=D10
      DO 5 I=1,N
5     D0(I)=D(I)
6     CONTINUE
      DO 7 I=1,N
      DO 7 J=1,N
      IF(B(I,J).GT.G) GOTO 7
      B(I,J)=0.0
      Q(I,J)=CM
7     CONTINUE
      GOTO 3
12    CONTINUE
      RETURN
      END
```

3.9.5 例　题

1）已知以下两个有向图的完好概率矩阵与运输容量矩阵。分别求出各图上顶点间的最大期望容量路。图中，已标出了各弧的完好概率值以及运输容量值。

（1）见图3.40。

$N=5$，$CM=90.0$

$$P(I,J)=\begin{pmatrix} 1.0 & 0.8 & 90.0 & 90.0 & 0.2 \\ 0.4 & 1.0 & 0.7 & 90.0 & 90.0 \\ 90.0 & 0.8 & 1.0 & 0.5 & 90.0 \\ 0.6 & 90.0 & 90.0 & 1.0 & 0.4 \\ 0.3 & 90.0 & 0.7 & 0.5 & 1.0 \end{pmatrix}$$

$$C(I,J)=\begin{pmatrix} 90.0 & 2.0 & 0.0 & 0.0 & 10.0 \\ 3.0 & 90.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 6.0 & 90.0 & 3.0 & 0.0 \\ 8.0 & 0.0 & 0.0 & 90.0 & 5.0 \\ 2.0 & 90.0 & 8.0 & 5.0 & 90.0 \end{pmatrix}$$

（2）见图3.41。

$N=4$，$CM=90.0$

$$P(I,J)=\begin{pmatrix} 1.00 & 0.70 & 0.65 & 90.00 \\ 0.80 & 1.00 & 0.80 & 90.00 \\ 90.00 & 0.50 & 1.00 & 90.00 \\ 0.90 & 0.30 & 0.70 & 1.00 \end{pmatrix}$$

$$C(I,J)=\begin{pmatrix} 90.00 & 10.00 & 12.00 & 0.00 \\ 7.00 & 90.00 & 15.00 & 0.00 \\ 0.00 & 6.00 & 90.00 & 0.00 \\ 8.00 & 13.00 & 7.00 & 90.00 \end{pmatrix}$$

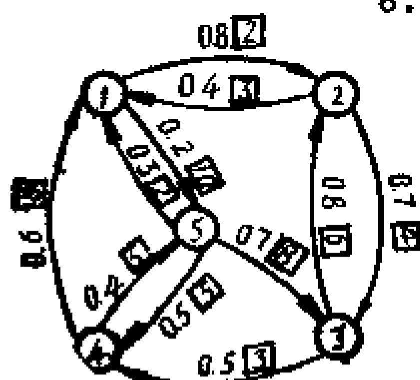


图 3.40

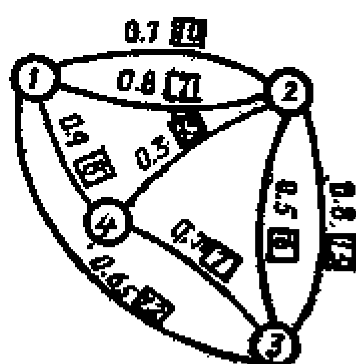


图 3.41

2）计算程序

```
      INTEGER D0(20)
      REAL P(20,20),C(20,20)
      WRITE(7,1)
      READ(5,*) N
      WRITE(7,2)
      READ(5,*) CM
1     FORMAT(1X,2HN=)
2     FORMAT(1X,3HCM=)
      CALL MAXCP(N,CM,P,C,D0)
      STOP
      END
C
      SUBROUTINE MAXCP(N,CM,P,C,D0)
      INTEGER S,T,D0(N),D(20)
      REAL P(N,N),C(N,N),Q(20,20)
      WRITE(7,3)
```

```
3       FORMAT(1X,7HP(I,J)=)
        DO 1 I=1,N
1       READ(5,5) (P(I,J),J=1,N)
5       FORMAT(20F8.2)
        WRITE(7,4)
4       FORMAT(1X,7HC(I,J)=)
        DO 7 I=1,N
7       READ(5,5) (C(I,J),J=1,N)
        WRITE(6,3)
        DO 6 I=1,N
        WRITE(6,2) (P(I,J),J=1,N)
6       CONTINUE
2       FORMAT(1X,20F8.2)
        WRITE(6,4)
        DO 8 I=1,N
        WRITE(6,2) (C(I,J),J=1,N)
8       CONTINUE
        DO 10 S=1,N
        DO 10 T=1,N
        IF (S.EQ.T) GOTO 10
        CALL MAXPC(N,CM,P,Q,D,C,D0,D00,S,T)
        WRITE(6,9)S,T,D00,(D0(I),I=1,N)
9       FORMAT(1X,2HS=,I2,1X,2HT=,I2,1X,
     #  2HF=,F8.2,4X,20I3)
10      CONTINUF
        RETURN
        END
```

3）计算结果

（1）

```
S=1  T=2  F=1.60    1 2
S=1  T=3  F=1.12    1 2 3
```

S = 1　T = 4　F = 0.56　1 2 3 4

S = 1　T = 5　F = 2.00　1 5

S = 2　T = 1　F = 1.20　2 1

S = 2　T = 3　F = 2.80　2 3

S = 2　T = 4　F = 1.05　2 3 4

S = 2　T = 5　F = 0.42　2 3 4 5

S = 3　T = 1　F = 0.96　3 2 1

S = 3　T = 2　F = 4.80　3 2

S = 3　T = 4　F = 1.50　3 4

S = 3　T = 5　F = 0.60　3 4 5

S = 4　T = 1　F = 4.80　4 1

S = 4　T = 2　F = 1.12　4 5 3 2

S = 4　T = 3　F = 1.40　4 5 3

S = 4　T = 5　F = 2.00　4 5

S = 5　T = 1　F = 1.50　5 4 1

S = 5　T = 2　F = 3.36　5 3 2

S = 5　T = 3　F = 5.60　5 3

S = 5　T = 4　F = 2.50　5 4

(2)

S = 1　T = 2　F = 7.00　1 2

S = 1　T = 3　F = 7.80　1 3

S = 1　T = 4　F = 0.00　0

S = 2　T = 1　F = 5.60　2 1

S = 2　T = 3　F = 12.00　2 3

S = 2　T = 4　F = 0.00　0

S = 3　T = 1　F = 2.40　3 2 1

S = 3　T = 2　F = 3.00　3 2

S = 3　T = 4　F = 0.00　0

S = 4　T = 1　F = 7.20　4 1

S = 4　T = 2　F = 5.04　4 1 2

S = 4　T = 3　F = 4.90　4 3

4. 独立集与支配集的计算

4.1 求图的全部极大独立集的布尔代数算法

4.1.1 功　能

我们以$G(V,E)$表示一个图，其中V为该图的顶点集合，E为该图的弧集合。所谓图的数立集，指的是顶点集V的一个子集合，要求这个子集合当中的任意两个顶点都不邻接。一个图可能会有若干个不同的数立集。我们考虑的是独立集当中称为极大独立集的子集合。只要给这个数立集增加任何一个属于该图G的顶点都会破坏该子集的独立性质，称这样的独立集为极大独立集。一般来说，图G所包含的极大独立集不一定唯一，这些不相同的极大独立集所包含的顶点数目也不一定相等。本节提供的算法可求出图的全部极大独立集。

4.1.2 方法概述

这儿使用的是布尔代数算法。利用布尔运算的方法与合并化简的规则，在运算过程中不断合并化简乘积布尔表达式数子项，从而节省存贮量，减少计算量。

对给定的图$G(V,E)$，$V=\{v_1,v_2,\cdots,v_N\}$为N个顶点的集合；$E=\{e_1,e_2,\cdots,e_M\}$为M条弧的集合。如果e_l的两端顶点为v_i与v_j，我们把v_i与v_j当作布尔量，则$e_l=v_i\wedge v_j$是布尔量v_i与v_j的布尔积。以布尔表达式$\varphi=\bigvee_{e_l\in E}(v_i\wedge v_j)$表示图$G$上所有弧的布尔和。对$\varphi$求“反”，记作$\overline{\varphi}$，

$$\overline{\varphi}=\bigwedge_{e_l\in E}(\overline{v_i}\vee\overline{v_j})。$$

经过整理，$\overline{\varphi}$可以表示为另一种形式：

$$\overline{\varphi}=\varphi_1\vee\varphi_2\vee\cdots\vee\varphi_k$$

其中：
$$\varphi_r=\overline{v_{r_1}}\wedge\overline{v_{r_2}}\wedge\cdots\wedge\overline{v_{r_j}}$$

$$r=1,2,\cdots,k。$$

由独立集的定义可以知道，极大独立集不能包含任何一条弧的两端的顶点。因此，显然有$\varphi=0$与$\overline{\varphi}=1$。只要在$\varphi_1,\varphi_2,\cdots,\varphi_k$之一$\varphi_i=1$，$1\leqslant i\leqslant k$；就有$\overline{\varphi}=1$也就得到$\varphi=0$。所以解$\varphi=0$的问题可以化为解$\varphi_i=0$，$i=1,2,\cdots,k$的问题。

在进行布尔运算中，充分利用了以下法则：

1. $v_i\vee v_i\wedge v_j=v_i$；
2. $v_i\wedge(v_i\vee v_j)=v_i$；
3. 当$r\geqslant0$，下式的前S个布尔量相同时，有

$$(v_{i_1}\vee v_{i_2}\vee\cdots\vee v_{i_s})\wedge(v_{i_1}\vee v_{i_2}\vee\cdots\vee v_{i_s}\vee v_{i_{s+1}}\vee\cdots\vee v_{i_{s+r}})$$
$$=v_{i_1}\vee v_{i_2}\vee\cdots\vee v_{i_s}$$

假定图$G(V,E)$有M条弧和N个顶点。经过编号的全体弧的两端顶点序号存放在数组$V(M,2)$当中。算法步骤如下：

步骤1 输入弧集$V(M,2)$。取第一条弧的两个顶点号，存入数组$T(M,N)$的前二行的与顶点序号相同的列内。该T数组的其余分量暂时存放零值。令$k=2$。

步骤2 取第k条弧的两端顶点序号，与数组T中各行元素进行逻辑乘运算，将布尔积送入数组$S(M,N)$内。

按照数组S各行包含顶点序号的数目多少，重新排列S各行的顺序，排序后的结果仍然存放在数组S内。

按照布尔运算法则删除S的部分行内容（置零），从而化简S。

将S的非全0的行排列紧凑，送到数组T内存放。令

$$k+1\Rightarrow k。$$

步骤3 判断是否$k\leqslant M$？若$k\leqslant M$，返回步骤2。否则，对T数组的全部非全0的行求反（所有的0换为1，所有的1换成0），就得到了图的全部最大独立集。计算完毕。

4.1.3 子程序参数说明

子程序名称 GMAXGL(M,N,V,G,S0)

M：图所含弧的数目，整型变量。

N：图的顶点数目，整型变量。

V(M,2)：图上所有弧的两端顶点序号数组，输入数据，整型数组。

G(40,N)：图的极大独立集的计算结果，整型数组。存放极大独立集的顶点序号。

其中：

$$G(i,j)=\begin{cases}0，若第\ i\ 个独立集不包含第\ j\ 号顶点；\\ j，若第\ i\ 个独立集包含第\ j\ 号顶点。\end{cases}$$

$i=1,2,\cdots,S0$; $j=1,2,\cdots,N$。

S0：图包含的极大独立集总数，输出结果，整型变量。

注：求极大独立集的算法占用工作单元较多。使用本子程序时可以根据需要、并考虑到计算机内存允许容量，适当扩大常界数组的尺寸。

4.1.4 子 程 序

```
      SUBROUTINE GMAXGL(M,N,V,G,S0)
      INTEGER V(M,2),G(40,N),T(70,31),
     # S(70,31),S0
      S0=0
      N1=N+1
      DO 1 I=1,70
      DO 1 J=1,N1
1     T(I,J)=0
      DO 2 I=1,N
      IF (V(I,1).EQ.I) T(1,I)=I
      IF (V(I,2).EQ.I) T(2,I)=I
2     CONTINUE
      T(1,N1)=1
      T(2,N1)=1
      L0=2
      DO 12 K=2,M
      K1=V(K,1)
      K2=V(K,2)
      DO 3 I=1,70
      DO 3 J=1,N1
3     S(I,J)=0
      J=0
```

```
         DO 5 I=1,L0
         IF (T(I,N1).EQ.0) GOTO 5
         J=J+1
         DO 4 L=1, N1
         S(J,L)=T(I,L)
4        S(J+1,L)=T(I,L)
         IF (S(J,K1).EQ.K1) GOTO 15
         S(J,K1)=K1
         S(J,N1)=S(J,N1)+1
15       J=J+1
         IF (S(J,K2).EQ.K2) GOTO 5
         S(J,K2)=K2
         S(J,N1)=S(J,N1)+1
5        CONTINUE
         DO 22 I=2,J
         JJ=I-1
         J1=0
         IF (S(JJ,N1)-S(I,N1)) 21,22,22
24       IF (S(JJ,N1).GE.S(I,N1)) GOTO 23
21       J1=J1+1
         JJ=JJ-1
         IF (JJ.GT.0) GOTO 24
23       DO 27 I1=1,N1
         IX=S(I,I1)
         JJ=I
26       JJ=JJ-1
         IF (JJ.LT.I-J1) GOTO 28
         S(JJ+1,I1)=S(JJ,I1)
         GOTO 26
28       II=I-J1
27       S(II,I1)=IX
```

```
22      CONTINUE
        J0=J-1
        DO 7 I=1,J0
        I1=I+1
        DO 6 L=I1,J
        DO 8 II=1,N
        IF (S(L,II).GT.S(I,II)) GOTO 6
8       CONTINUE
        DO 9 II=1,N1
9       S(I,II)=0
        GOTO 7
6       CONTINUE
7       CONTINUE
        L=0
        DO 10 I=1,J
        IF (S(I,N1).EQ.0) GOTO 10
        L=L+1
        DO 11 II=1,N1
11      T(L,II)=S(I,II)
10      CONTINUE
        L0=L
        S0=L
12      CONTINUE
        DO 13 I=1,40
        DO 13 J=1,N
        G(I,J)=0
        IF (I.GT.L0) GOTO 13
        IF (T(I,J).EQ.0) G(I,J)=J
13      CONTINUE
        RETURN
        END
```

4.1.5 例　题

1）求两个图的全部极大独立集，并根据每个极大独立集作图。

（1）见图4.1。

$M=8$，$N=6$

$$V(I,J)=\begin{pmatrix}1&2\\1&3\\1&4\\2&4\\3&4\\4&5\\4&6\\5&6\end{pmatrix}$$

（2）见图4.2。

$M=7$　$N=7$

$$V(I,J)=\begin{pmatrix}1&2\\2&3\\3&4\\3&5\\4&5\\4&6\\6&7\end{pmatrix}$$

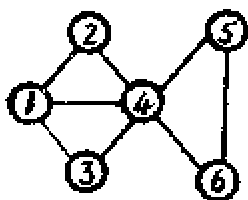


图　4.1

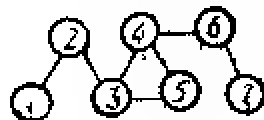


图　4.2

2）计算程序

```
INTEGER V(40,2),G(40,20)
```

```
      WRITE(7,1)
      READ(5,*) M
      WRITE(7,2)
      READ(5,*) N
      CALL GM(M,N,V,G)
1     FORMAT(1X,2HM=)
2     FORMAT(1X,2HN=)
      STOP
      END
C
      SUBROUTINE GM(M,N,V,G)
      INTEGER V(M,2),G(40,N),S0
      WRITE(7,1)
1     FORMAT(1X,7HV(I,J)=)
      READ(5,*) (V(I,1),V(I,2),I=1,M)
      WRITE(6,3) N,M
3     FORMAT(1X,'N=',I2,'   M=',I2,/)
      CALL GMAXGL(M,N,V,G,S0)
      WRITE(6,2)S0
2     FORMAT(1X,'S0=',I2)
      WRITE(6,13)
13    FORMAT(1X,'G(I,J)=',/)
      DO 4 I=1,S0
      WRITE(6,5) I,(G(I,J),J=1,N)
5     FORMAT(1X,'I=',I2,4X,20I3)
      RETURN
      END
```

3）计算结果

（1） S0=5

$$G(I,J)=\begin{pmatrix} 0&0&0&4&0&0\\ 1&0&0&0&0&6\\ 1&0&0&0&5&0\\ 0&2&3&0&0&6\\ 0&2&3&0&5&0 \end{pmatrix}$$

将全部极大独立集作图4.3。

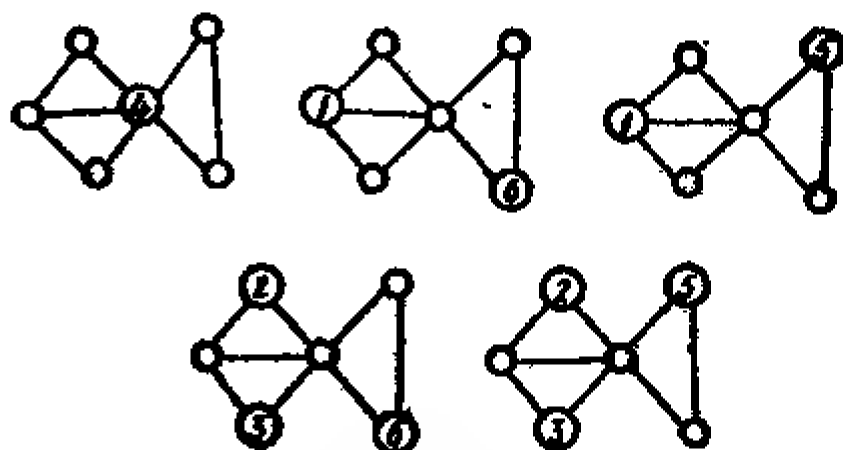


图 4.3

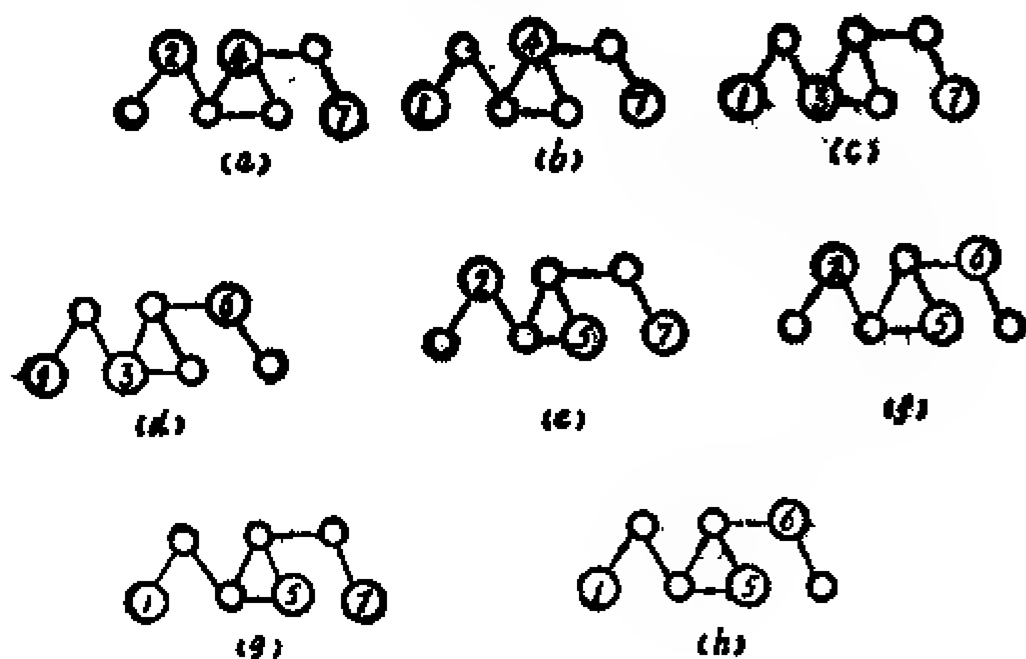


图 4.4

（2） S0=8

$$G(I,J)=\begin{pmatrix} 0&2&0&4&0&0&7\\ 1&0&0&4&0&0&7\\ 1&0&3&0&0&0&7\\ 1&0&3&0&0&6&0\\ 0&2&0&0&5&0&7\\ 0&2&0&0&5&6&0\\ 1&0&0&0&5&0&7\\ 1&0&0&0&5&6&0 \end{pmatrix}$$

将全部极大独立集作图4.4。

4.2 求图的全部极小支配集的布尔代数算法

4.2.1 功　能

对于图$G(V,E)$，满足如下条件的顶点子集合称为图G的支配集：该图中任何顶点或者属于该子集，或者与该子集中的一个顶点相邻。可以看出，一个图的支配集是很多的。在此，我们仅考虑十分有用的极小支配集。对于一个支配集，如果去了它的任何一个顶点，它都不再是支配集了，称这一类支配集为极小支配集。支配集也称为控制集。由上述极小支配集的定义可知，图的极大独立集都是它的极小支配集，反过来却不一定成立。极小支配集不一定是唯一的，而且各极小支配集所包含的顶点数目不一定相同。

本节给出一个求图的全部极小支配集的布尔代数算法。

4.2.2 方法概述

本节采用的算法与上一节求图的全部极大独立集的布尔代数算法基本相同。

图$G(V,E)$有N个顶点分别用布尔变量V_1，$V_2\cdots$，

V_N来表示。以布尔表达式φ_i表示顶点V_i与和它相邻的全部顶点组成的顶点子集合。则

$$\varphi_i = v_i \vee (\bigvee_{v_j \in Ad_j(v_i)} v_j) \quad i=1,2,\cdots,N。$$

上式中用$Ad_j(v_i)$表示与顶点v_i相邻的顶点子集。令

$$\psi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \cdots \wedge \varphi_N$$

布尔表达式ψ就是图G的全部极小支配集。再利用布尔运算法则：

$$v_i \vee v_j = v_i = v_j,$$

$$v_i \vee v_i \wedge v_j = v_i,$$

$$(v_{i_1} \vee v_{i_2} \vee \cdots \vee v_{i_r}) \wedge (v_{i_1} \vee v_{i_2} \vee \cdots \vee v_{i_r} \vee v_{i_{r+1}} \vee \cdots \vee v_{r+s})$$

$$= v_{i_1} \vee v_{i_2} \vee \cdots \vee v_{i_r} \qquad 当 S \geqslant 0 时；$$

经过整理与化简，将ψ化为如下的布尔和式

$$\psi = \psi_1 \vee \psi_2 \vee \psi_3 \vee \cdots \vee \psi_{s0}$$

其中：$\psi_i = v_{i_1} \wedge v_{i_2} \wedge v_{i_3} \wedge \cdots \wedge v_{i_r}$，$i=1,2,\cdots,S_0$。$\psi$是一个极小支配集。这样，就找到了图$G$的全部极小支配集。

为了计算的方便，用$V(N,N)$矩阵记录图上N个顶点间的邻接关系。规定：

$$V(i,j)=\begin{cases} j，如果第 i 点与第 j 点相邻；\\ 0，如果第 i 点与第 j 点不相邻。\end{cases}$$

$$i,j=1,2,\cdots,N。$$

为了保证结集单元的存贮量，以$M0$表示图的全部最小支配集数目的上界值。

步骤1 输入图的顶点集目N，全部最小支配集的预计上界值$M0$，以及$V(N,N)$。

将$V(N,N)$传送入$V0(N,N+1)$的各行前N列。在$V0$

的各行最后一列存放每行非 0 元素的个数。

将$V0(N,N+1)$各行按照每行第$N+1$列值的大小下降顺序重新排列。然后对各行内容按照布尔运算法则进行删减，经删减的空行作压缩排列处理。

将$V0(N,N+1)$的第 1 行传送入$T(40,N+1)$，令$k=2$。

步骤 2　取$V0(N,N+1)$的第k行与T内各行进行布尔乘法运算。然后对T按第$N+1$列的降序重排与删除、压缩，结果仍然存于$V0$内。
令$k=k+1$。

步骤 3　$V0$当中如果还有尚未与T作布尔乘运算的行，则返回步骤 2；否则，将T内存放的图的全部最小支配集的计算结果送$G(M0,N)$。计算结束。

程序中多次用到"按第$N+1$列降序重排"删除与压缩的处理。目的是利用布尔运算法则去简化布尔乘法的结果项。这样可以大大减少重复贮存与以后的重复布尔运算。这段工作由一个单独的子程序PASG来完成。

4.2.3 子程序参数说明

1）求全部极小支配集子程序名称 GMINGL（N,M0,V,G,S0）

N：图的顶点数目，整型变量。

M0：预计极小支配集数目上界，输入数据，整型变量。

V（N,N）：图上各顶点邻接关系输入数据，整型数组。其中：

$$V(i,j)=\begin{cases} j, & \text{当第 } i \text{ 顶点与第 } j \text{ 顶点相邻;} \\ 0, & \text{当第 } i \text{ 顶点与第 } j \text{ 顶点不相邻。} \end{cases}$$

$i=1,2,\cdots,N$; $j=1,2,\cdots,V$。

G0（M0,N）：全部极小支配集计算结果的输出数据，整型数组。其中：

$$G(i,j)=\begin{cases} j, & \text{当 } i \text{ 个极小支配集包含第 } j \text{ 顶点；} \\ 0, & \text{当 } i \text{ 个极小支配集不包含第 } j \text{ 顶点。} \end{cases}$$

$i=1,2,\cdots,S0$; $j=1,2,\cdots,N$。

$G(i,j)=0$，当$i>S0$时，$j=1,2,\cdots,N$。

S0：全部极小支配集的总数，输出结果，整型变量。

注：① 要求M0的预计值不小于最终算出的$S0$，否则$G(M0,N)$的行数过于小，发生丢失极小支配集的情况。

② 程序中的工作单元数组采用了常界数组，可以根据调用时的需要，并考虑到计算机的容量进行修改。但是在修改中应当注意到与子程序中循环语句的循环上界要统一修改。

2）排序、删除与压缩子程序名称PASG(M,N0, N1, N2,N3,U)

M：指定的按降序重排顺序的列顺序号，整型变量。

N0：被排序数组需要排序的行数，整型变量。

N1：被排序数组需要排序的列数，整型变量

N2：被排序数组的总行数，整型变量。要求$N2\geqslant N0$。

N3：被排序数组的总列数，整型变量。要求$N3\geqslant N1$。

U(N2,N3)：被排序、删除、压缩的数组名、整型数组。

4.2.4 子 程 序

```
      SUBROUTINE GMINGL (N,M0,V,G,S0)
      INTEGER S0,V(N,N),G(M0,N),S(40,21) .
   #  T (40,21), V0(20,21), Q(21)
      N1=N+1
      DO 1 I=1,40
      DO 1 J=1,N1
```

```
      T(I,J)=0
1     S (I,J)=0
      DO 3 I=1,N
      V0(I,N1)=0
      DO 2 J=1,N
      IF(V(I,J).NE.0) V0(I,N1)=V0(I,N1)
    # +1
2     V0(I,J)=V(I,J)
3     CONTINUE
      CALL PASG(N1,N,N2,20,21,V0)
      N0=0
      DO 4 I=1,N
      IF(V0(I,N1).GT.0)N0=N0+1
4     CONTINUE
      L0=V0(1,N1)
      J0=1
      DO 14 I=1,L0
      DO 5 J=J0,N
      IF(V0(1,J).EQ.0) GOTO 5
      T(I,J)=J
      T(I,N1)=1
      J0=J+1
      GOTO 14
5     CONTINUE
14    CONTINUE
      DO 12 K=2,N0
      IF(V0(K,N1).EQ.0) GOTO 12
      DO 6 J=1, N1
6     Q(J)=V0(K,J)
      L=0
      DO 7 J=1,N
```

```
      IF(Q(J).EQ.0)GOTO 7
      DO 9  I1=1, L0
      DO 8 J1=1,N1
8     S(I1+L,J1)=T(I1,J1)
      IF (S(I1+L,J).EQ.J)GOTO 9
      S(I1+L,J)=J
      S(I1+L,N1)=S(I1+L,N1)+1
9     CONTINUE
      CALL PASG(N1,40,N1,40,21,S)
      L1=0
      DO 10 I=1,40
      IF (S(I,N1).NE.0) L1=L1+1
10    CONTINUE
      L=L1
7     CONTINUE
      L0=L
      DO 11 I1=1,L0
      DO 11 J1=1,N1
11    T(I1,J1)=S(I1,J1)
12    CONTINUE
      S0=L0
      DO 13 I=1,M0
      DO 13 J=1,N
13    G(I,J)=T(I,J)
      RETURN
      END
C
      SUBROUTINE PASG(M,N0,N1,N2,N3,U)
      INTEGER U(N2,N3)
      N=M-1
      DO 2 I=2, N0
```

```
      J=I-1
      J1=0
      IF(U(J,M)-U(I,M)) 1,2,2
4     IF(U(J,M).GE.U(I,M))GOTO 3
1     J1=J1+1
      J=J-1
      IF(J.GT.0)GOTO 4
      DO 7 I1=1,N1
3     IX=U(I,I1)
      J=I
6     J=J-1
      IF(J.LT.I-J1) GOTO 8
      U(J+1,I1)=U(J,I1)
      GOTO 6
8     II=I-J1
7     U(II,I1)=IX
2     CONTINUE
      J0=N0-1
      DO 17 I=1,J0
      IF(U(I,M).EQ.0) GOTO 17
      I1=I+1
      DO 16 L=I1,N0
      IF(U(L,M).EQ.0) GOTO 16
      DO 18 II=1,N
      IF(U(L,II).GT. U(I,II)) GOTO 16
18    CONTINUE
      DO 19 II=1,N1
19    U(I,II)=0
      GOTO 17
16    CONTINUE
17    CONTINUE
```

```
      N=0
      DO 21 I=1,N0
      IF(U(I,M).NE.0) N=N+1
21    CONTINUE
      DO 22 I=1,N
      L=0
      IF(U(I,M).NE.0) GOTO 22
23    L=L+1
      IF(U(I+L,M).EQ.0) GOTO 23
      DO 24 J=1,M
      U(I,J)=U(I+L,J)
24    U(I+L,J)=0
22    CONTINUE
      RETURN
      END
```

4.2.5 例　题

1）求以下三个图的全部极小支配集。

（1）见图4.5。

$N=6, M0=8$

$$V(I,J)=\begin{pmatrix} 1 & 2 & 3 & 4 & 0 & 0 \\ 1 & 2 & 0 & 4 & 0 & 0 \\ 1 & 0 & 3 & 4 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 4 & 5 & 6 \\ 0 & 0 & 0 & 4 & 5 & 6 \end{pmatrix}$$

（2）见图4.6。

$N=9,\ M0=20$

$$V(I,J)=\begin{vmatrix} 1 & 2 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 6 & 0 & 0 & 0 \\ 1 & 0 & 0 & 4 & 5 & 0 & 7 & 0 & 0 \\ 0 & 2 & 0 & 4 & 5 & 6 & 0 & 8 & 0 \\ 0 & 0 & 3 & 0 & 5 & 6 & 0 & 0 & 9 \\ 0 & 0 & 0 & 4 & 0 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 7 & 8 & 9 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 8 & 9 \end{vmatrix}$$

（3）见图4.7。

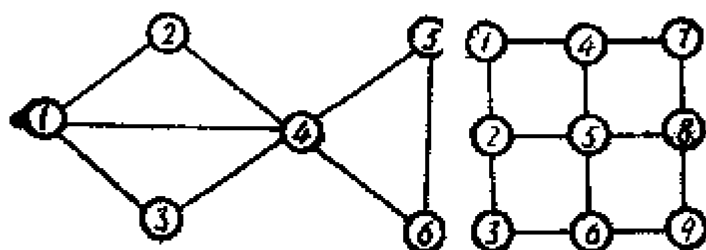


图 4.5

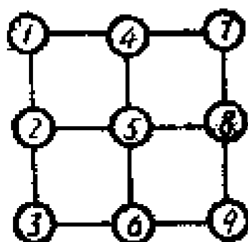


图 4.6

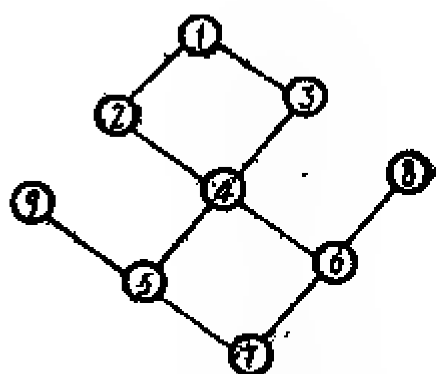


图 4.7

$N=9$，$M0=25$

$$V(I,J)=\begin{vmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 4 & 5 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 & 0 & 7 & 0 & 9 \\ 0 & 0 & 0 & 4 & 0 & 6 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 9 \end{vmatrix}$$

2）计算程序

```
      INTEGER S0,V(20,20),G(50,20)
      WRITE(7,1)
1     FORMAT(1X,2HN=)
      READ(5,*)N
      WRITE(7,2)
2     FORMAT(1X,3HM0=)
      READ(5,*)M0
      CALL GMIN(N,M0,V,G,S0)
      STOP
      END
C
      SUBROUTINE GMIN(N,M0,V,G,S0)
      INTEGER S0,V(N,N),G(M0,N)
      WRITE(7,1)
1     FORMAT(1X,7HV(I,J)=)
      READ(5,*)((V(I,J),J=1,N),I=1,N)
      WRITE(6,5)
5     FORMAT(1X,'V(I,J)=',/)
      DO 6 I=1,N
6     WRITE(6,7)(V(I,J),J=1,N)
7     FORMAT(1X,20I3)
      CALL GMINGL(N,M0,V,G,S0)
      WRITE(6,2)S0
2     FORMAT(1X,3HS0=,I4,/,1X,7HG(I,J)=)
      DO 3 I=1,S0
3     WRITE(6,7) (G(I,J),J=1,N)
      RETURN
      END
```

3）计算结果

（1） S0=5

$$
G(I,J)=\begin{pmatrix}
0 & 2 & 3 & 0 & 5 & 0 \\
0 & 2 & 3 & 0 & 0 & 6 \\
1 & 0 & 0 & 0 & 5 & 0 \\
1 & 0 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & 4 & 0 & 0
\end{pmatrix}
$$

见图4.8。

（2） S0=16

$$
G(I,J)=\begin{pmatrix}
1 & 0 & 3 & 0 & 5 & 0 & 7 & 0 & 9 \\
0 & 0 & 3 & 4 & 0 & 6 & 7 & 0 & 0 \\
0 & 2 & 0 & 4 & 0 & 6 & 0 & 8 & 0 \\
0 & 2 & 3 & 0 & 0 & 0 & 7 & 8 & 0 \\
1 & 0 & 0 & 4 & 0 & 6 & 0 & 0 & 9 \\
1 & 2 & 0 & 0 & 0 & 0 & 0 & 8 & 9 \\
0 & 0 & 0 & 4 & 5 & 6 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 6 & 7 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 6 & 7 & 0 & 0 \\
1 & 0 & 3 & 0 & 0 & 0 & 0 & 8 & 0 \\
0 & 0 & 3 & 4 & 0 & 0 & 0 & 8 & 0 \\
1 & 0 & 0 & 0 & 0 & 6 & 0 & 8 & 0 \\
0 & 2 & 0 & 0 & 5 & 6 & 0 & 8 & 0 \\
0 & 2 & 0 & 4 & 0 & 0 & 0 & 0 & 9 \\
0 & 0 & 3 & 4 & 0 & 0 & 0 & 0 & 9 \\
0 & 2 & 0 & 0 & 0 & 0 & 7 & 0 & 9
\end{pmatrix}
$$

见图4.9。

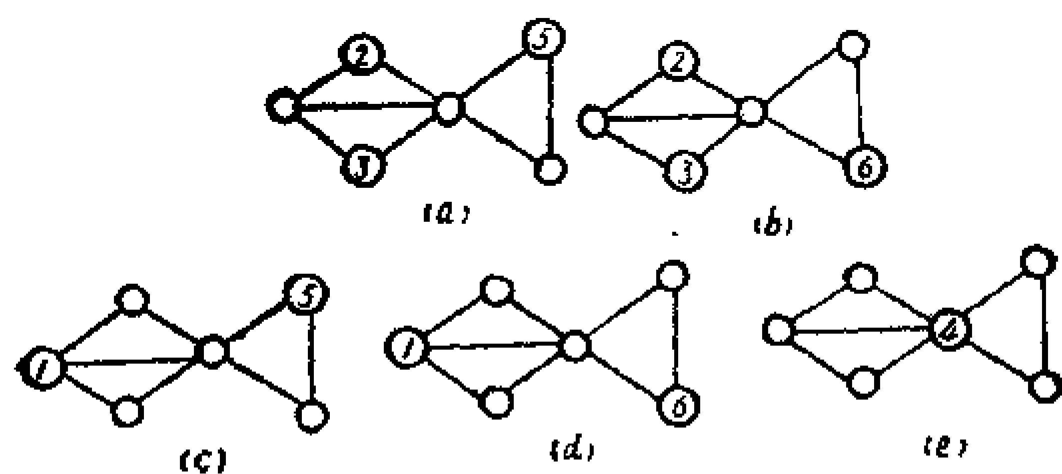


图 4.8

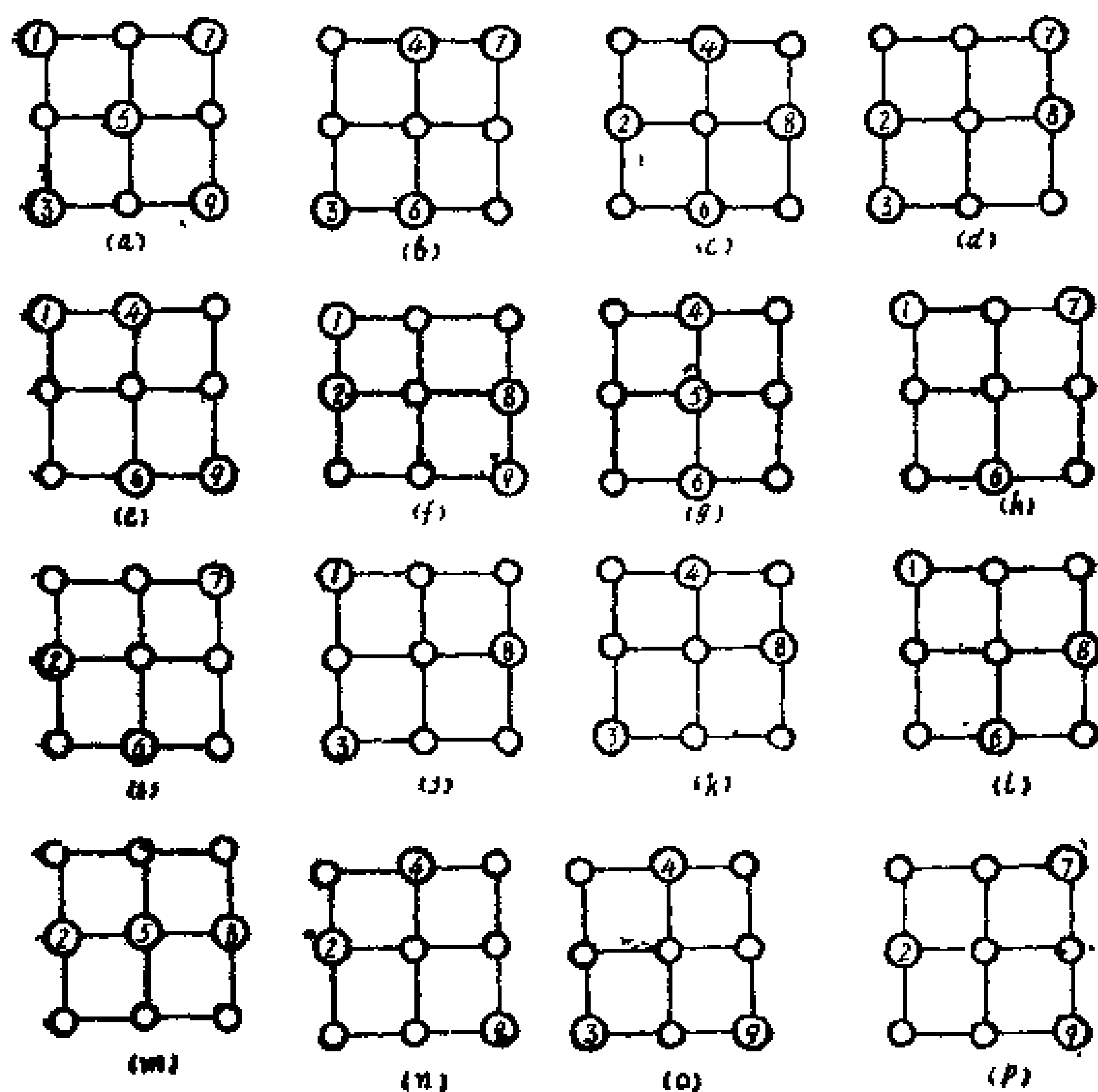


图 4.9

（3）$S_0=18$

$$
G(I,J)=\begin{pmatrix}
1 & 2 & 0 & 0 & 0 & 0 & 7 & 8 & 9\\
1 & 0 & 3 & 0 & 0 & 0 & 7 & 8 & 9\\
0 & 2 & 3 & 0 & 0 & 0 & 7 & 8 & 9\\
1 & 0 & 0 & 4 & 0 & 0 & 7 & 8 & 9\\
0 & 0 & 3 & 4 & 0 & 0 & 7 & 8 & 9\\
0 & 2 & 0 & 4 & 0 & 0 & 7 & 8 & 9\\
0 & 2 & 3 & 0 & 5 & 0 & 0 & 8 & 0\\
0 & 0 & 3 & 4 & 5 & 6 & 0 & 8 & 0\\
0 & 2 & 0 & 4 & 5 & 0 & 0 & 8 & 0\\
0 & 2 & 3 & 0 & 5 & 6 & 0 & 0 & 0\\
0 & 0 & 3 & 4 & 5 & 6 & 0 & 0 & 0\\
0 & 2 & 0 & 4 & 5 & 6 & 0 & 0 & 0\\
0 & 2 & 3 & 0 & 0 & 6 & 0 & 0 & 9\\
0 & 0 & 3 & 4 & 0 & 6 & 0 & 0 & 9\\
0 & 2 & 0 & 4 & 0 & 6 & 0 & 0 & 9\\
1 & 0 & 0 & 0 & 5 & 0 & 0 & 8 & 0\\
1 & 0 & 0 & 0 & 5 & 6 & 0 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 9
\end{pmatrix}
$$

极小支配集的作图从略。

5. 匹配的计算

5.1 二分图最大基数匹配的算法 I

5.1.1 功　能

二分图也称为偶图。如果该二分图的一个弧集合用M表示，若原图的各顶点最多与集合M当中的一条弧关联，则称集合M为一个匹配。一个匹配包含弧的数目称为匹配的基数。本节给出一个求二分图最大基数匹配的程序。这类问题在生产实践中有着广泛的应用价值。例如，对$N1$个待分配人员安排工作的问题，有$N2$种工作可以对他们进行安排。由于各人条件不同，他们每人分别在这$N2$项工作中有几项适合本人能力。如何分配才能使尽可能多的人得到一项工作？这就是一个求二分图的最大基数匹配的问题。

5.1.2 方法概述

二分图G的顶点可以分为含$N1$个顶点的子集X与含$N2$个顶点的子集$\overline{Y}$。$N1+N2=N$是图G的顶点总数。X集内部的顶点互不邻接，$\overline{Y}$集内部的顶点也互不邻接。因此，图G中的任意一条弧的两个顶点都分别属于X集与$\overline{Y}$集。在使用本节的算法之前，首先应当检查一下需要计算匹配的图是不

是二分图。

在这里用到的方法，是由一个初始匹配出发，逐次进行调整，不断扩大匹配的基数，一直到无法进一步扩大匹配的基数为止，就得到了图G的一个最大基数匹配方案。为达到这个目的，应当事先输入一个X与$\overline{Y}$各顶点间的邻接关系矩阵$A(N1,N2)$。其中：

$$A(i,j)=\begin{cases}1，当X中的第\ i\ 顶点与Y中的第\ j\ 顶点邻接时；\\0，当X中的第\ i\ 顶点与\overline{Y}中的第\ j\ 顶点不邻接时。\end{cases}$$

$$i=1,2,\cdots,N1;\quad j=1,2,\cdots,N2。$$

为了减少扩大匹配时的检查次数，不妨令$N1<N2$。

本算法的具体执行步骤如下：

步骤1　对邻接矩阵A进行扫描，给出一个基数较大的初始匹配，记作M。转步骤2。

步骤2　将X与Y两个顶点集中的不与匹配M的弧相关联的所有顶点都称为暴露顶点。由X当中未检查过的一个暴露顶点x_0出发，生长一棵M-交错树。随着M-交错树的生长，对交错树上的顶点记下标号（实际上是记下它的在M-交错树上前一个顶点的序号）。当这棵交错棵生长到集合Y当中的一个暴棵顶点（记作y_0）时，就存在一条由x_0至y_0的可扩充路。转向步骤3。

如果M-交错树已经无法继续生长，仍然未生长到$\overline{Y}$当中的暴露顶点，则找不到由x_0出发的可扩充路。转步骤4。

步骤3　对找到的M-交错树，由y_0向x_0的记标顺序进行反向追踪，得到由x_0至y_0所经过顶点的顺序号。将这条可扩充路中原先已属于匹配的弧从匹配M当中去掉。而将原先不属于匹配的弧记入到匹配M。M中不属于可扩充路的弧不

变。这样就形成了增加一个基数的新匹配方案，仍然记作M。转步骤4。

步骤4　若集X中还有未检查过的暴露顶点，则转向步骤2；否则结束。

5.1.3 子程序参数说明

子程序名称　MAXG2(N1,N2,A,D,K0)

N1：集合X的顶点数目，整型变量。

N2：集合$\overline{Y}$的顶点数目，宜取$N1 \leqslant N2$；否则，可以交换X与$\overline{Y}$，整型变量。

A(N1,N2)：集合X与集合$\overline{Y}$上各顶点间的邻接关系矩阵，输入数据，整型数组。其中：

$$A(i,j)=\begin{cases}1, & \text{当}X\text{中第 }i\text{ 顶点与}\overline{Y}\text{中第 }j\text{ 顶点邻接时；}\\ 0, & \text{当}X\text{中第 }i\text{ 顶点与}\overline{Y}\text{中第 }j\text{ 顶点不邻接时。}\end{cases}$$

$i=1,2,\cdots,N1$；$j=1,2,\cdots,N2$。

D(N1,2)：该图最大基数匹配的计算结果存放的单元，整型数组。其中$D(i,1)$与$D(i,2)$分别存放最大基数匹配中的第i条弧的属于X与属于$\overline{Y}$的顶点序号。如果有$K0<N1$，当$i>K0$时，有$D(i,1)=D(i,2)=0$。

K0：图的最大基数匹配的基数，输出结果，整型变量。

5.1.4 子程序

```
SUBROUTINE MAXG2(N1,N2,A,D,K0)
```

```
        INTEGER A(N1,N2),D(N1,2),U,V,S1(25).
        ,S2(25),S3(25),S4(25),S(50)
        DO 1 I=1,N1
        D(I,1)=0
        D(I,2)=0
 1      S1(I)=0
        DO 2 I=1,N2
 2      S2(I)=0
        K0=0
        DO 4 I=1,N1
        DO 5 J=1,N2
        IF(A(I,J).EQ.0.OR.S1(I)+S2(J)
     #  .NE.0) GOTO 5
        D(I,1)=I
        D(I,2)=J
        S1(I)=1
        S2(I)=1
        K0=K0+1
        GOTO 4
 5      CONTINUE
 4      CONTINUE
        DO 12 I=1,N1
        IF(S1(I).EQ.1) GOTO 12
        DO 6 J=1,N1
 6      S3(J)=0
        DO 7 J=1,N2
 7      S4(J)=0
        V=0
        N3=2*N1
        DO 8 J=1,N3
 8      S(J)=0
```

```
      DO 9 J=1,N2
      IF (A(I,J).EQ.0) GOTO 9
      V=V+1
      S3(I)=-11111
      S4(J)=I
9     CONTINUE
      IF(V.EQ.0) GOTO 12
13    DO 30 J=1,N2
      IF(S4(J).EQ.0) GOTO 30
      DO 10 K=1, N1
      IF (A(K,J).EQ.1.AND. S3(K).EQ.0.AND.
    # D(K,2).EQ.J) S3(K)=J
10    CONTINUE
30    CONTINUE
      V=0
      DO 11 K=1,N1
      IF(S3(K).LE.0) GOTO 11
      DO 15 J=1,N2
      IF(A(K,J).EQ.1.AND.S2(J)
    # .EQ.0) GOTO 14
      IF (A(K,J).LT.1. OR.S4(J).NE.0.OR.
    # D(K,2).EQ.J) GOTO 15
      V=V+1
      IF(S3(K).GT.0) S3(K)=-S3(K)
      S4(J)=K
15    CONTINUE
11    CONTINUE
      IF(V.GT.0) GOTO 13
      GOTO 12
14    DO 16 V=1,N1
16    IF(S3(V).LT.0) S3(V)=-S3(V)
```

```
          S(1)=J
          S(2)=K
          U=3
25        CONTINUE
          IF (S3(K).EQ.11111) GOTO 26
          S(U)=S3(K)
          J=S(U)
          U=U+1
          S(U)=S4(J)
          K=S(U)
          U=U+1
          GOTO 25
26        KU=U-1
          DO 27 K=2,KU,2
          L=U+1-K
          V=S(L)
          D(V,1)=V
27        D(V,2)=S(L-1)
          K0=K0+1
12        CONTINUE
          RETURN
          END
```

5.1.5 例　题

1）用求二分图最大基数匹配的算法，计算以下问题。

（1）有六户要求分配住房的居民和六套待分配的房屋。已知各申请住房的居民（分别以x_i表示）各自在这六套住房（分别以y_j表示）当中选中了几套。凡是x_i选中了y_j者，在图上（见图5.1）将x_i与y_j用虚线连接起来。要求给

出一个使尽可能多的居民分得一套自己满意的住房的分配方案。

$N1=6$, $N2=6$

$$A(I,J)=\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

（2）有八件不同的工作供六个人选择分配（见图5.2）。已知这六个人分别适合完成八件工作当中的几种。给出一个使尽可能多的人员得到一件适合本人情况的工作的分配方案。

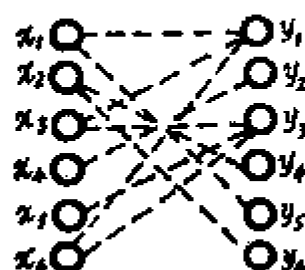


图 5.1

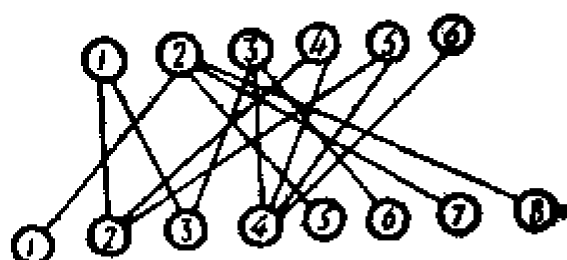


图 5.2

$N1=6$, $N2=8$

$$A(I,J)=\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2）计算程序

```
      INTEGER A(20,25),D(20,2)
      WRITE(7,1)
      READ(5,*) N1
      WRITE(7,2)
      READ(5,*) N2
1     FORMAT(1X,3HN1=)
2     FORMAT(1X,3HN2=)
      CALL MG2(N1,N2,A,D)
      STOP
      END
      SUBROUTINE MG2(N1,N2,A,D)
      INTEGER A(N1,N2), D(N1,2)
      WRITE(7,3)
3     FORMAT(1X,7HA(I,J)=)
      READ(5,*) ((A(I,J),J=1,N2),I=1,N1)
      WRITE(6,4) N1,N2
4     FORMAT(1X,'N1=',I2,4X,'N2=',I2,/)
      DO 5 I=1,N1
5     WRITE(6,1) (A(I,J), J=1,N2)
1     FORMAT(1X,25I3)
      CALL MAXG2(N1,N2,A,D,K0)
      WRITE(6,10)
10    FORMAT(1X,8X,'X Y')
      DO 6 I=1, K0
6     WRITE(6,7) I,D(I,1),D(I,2)
7     FORMAT(1X,'I=',I2,4X,I2,'—',I2)
      RETURN
      END
```

3）计算结果

(1) K0=5

	X	Y
I=1	1	-5
I=2	2	-4
I=3	3	1
I=4	4	-2
I=5	5	-3

见图5.3。

(2) K0=5

	X	Y
I=1	1	-3
I=2	2	-1
I=3	3	6
I=4	4	-4
I=5	5	-2

见图5.4。

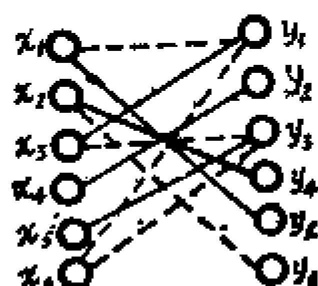


图 5.3

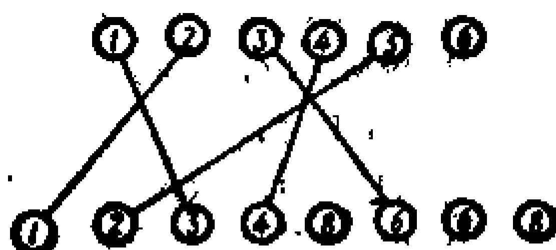


图 5.4

5.2 二分图最大基数匹配的算法Ⅱ

5.2.1 功　能

本节给出一个求二分图最大基数匹配的网络最大流算法。程序的结果与算法Ⅰ相同，得到一个基数最大的匹配方案。

5.2.2 方法概述

已知二分图的顶点可分为X与Y两个子集。图上的所有弧只能够联结X与Y的顶点，而不能够联结X或Y内各自的顶点。为了使用网络流方法进行计算，首先对该图作如下处理：

1）将每条弧给一个由集合X上的顶点到集合Y上的顶点的定向。

2）增加一个新设顶点S，并对所有属于集合X的顶点x增加一条有向弧（S,x）。

3）增加一个新设顶点T，并对所有属于集合Y的顶点y增加一条有向弧（y,T）。

4）令所有弧（包括原有弧与增设的弧）的允许流量为1。

经过以上变化，得到一个网络流图。我们只要使用计算网络最大流的子程序去求由出发点S至终点T的最大流。由最大流的结果中得到顶点集X与Y之间的一个最大基数匹配方案。以下给一个简单的二分图化为网络流图的例图（见图5.5）。

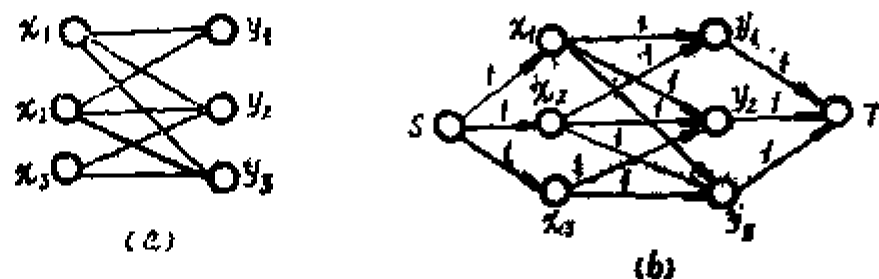


图 5.5

子程序只要求你输入原二分图的X与Y之间的顶点邻接矩阵。网络流量矩阵在程序的处理中自动产生。最后输出的

结果是最大基数匹配的结果。

5.2.3 子程序参数说明

子程序名称　MAXXY (N,M,NM,X,Z,Y,L,K)

N：属于顶点集X的顶点数目，整型变量。

M：属于顶点集Y的顶点数目，整型变量。

NM：工作单元数组上界，$NM=N+M+2$，整型变量。

X(N,M)：X与Y之间各顶点的邻接矩阵，当x_i与y_j邻接时数组元素$X(i,j)=1$；否则，$X(i,j)=0$。输入数据，整型数组。

$Z(N,2)$：图的最大基数匹配的输出结果。其中$Z(i,1)$与$Z(i,2)$分别存放匹配的第i条弧相关的两个顶点在X与Y中的顺序号。匹配数$K<N$时，$Z(j,1)=Z(j,2)=0, j=K+1,\cdots N$。整型数组。

Y(NM,NM)，L(NM,NM)：工作单元，整型数组。

K：最大基数匹配的基数，计算结果，整型变量。

注：本程序调用了网络最大流算法子程序MAXC。

5.2.4 子　程　序

```
      SUBROUTINE MAXXY(N,M,NM,X,Z,Y,L,K)
      INTEGER X(N,M),Z(N,2),Y(NM,NM),
     #  L(NM,NM)
      DO 1 I=1,NM
      DO 1 J=1,NM
1     Y(I,J)=0
```

```
      DO 2 J=1,N
2     Y(1,J+1)=1
      N2=N+2
      N0=NM-1
      DO 3 I=N2,N0
3     Y(I,NM)=1
      DO 4 I=1,N
      DO 4 J=1,M
4     Y(I+1,J+N+1)=X(I,J)
      M0=N
      N1=1
      CALL MAXC(NM,Y,L,N1,NM,M0,MC)
      DO 5 I=1,N
      Z(I,1)=0
5     Z(I,2)=0
      K=0
      DO 7 I=1,N
      DO 6 J=1,M
      IF (L(I+1,J+N+1).NE.1) GOTO 6
      K=K+1
      Z(K,1)=I
      Z(K,2)=J
6     CONTINUE
7     CONTINUE
      RETURN
      END
```

5.2.5 例　题

1）有六个准备安排工作的人与七项工作。这些人各自

能够胜任七项工作中的若干种。图5.6中，以x_i表示人员代号，以y_j表示工作种类代号，连接弧表示某人可以胜任的工作关系。用本节算法求一个使得到工作人数最多的分配方案。

$N=6$，$M=7$

$$X(I,J)=\begin{matrix}1&0&0&0&0&0&0\\1&1&0&1&0&0&0\\1&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&1&1&1&1\end{matrix}$$

2）计算程序

```
      INTEGER X(20,20),Z(20,2),Y(42,42),
   #  L(42,42)
      WRITE(7,1)
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)M
1     FORMAT(1X,2HN=)
2     FORMAT(1X,2HM=)
      NM=N+M+2
      CALL MXY(N,M,NM,X,Z,Y,L)
      STOP
      END
C
      SUBROUTINE MXY(N,M,NM,X,Z,Y,L)
      INTEGER X(N,M),Z(N,2),Y(NM,NM),
   #  L(NM,NM)
      WRITE(7,3)
3     FORMAT(1X,7HX(I,J)=)
```

```
      READ(5,*) ((X(I,J),J=1,M),I=1,N)
      CALL MAXXY(N,M,NM,X,Z,Y,L,K)
      WRITE(6,52)
52    FORMAT(1X,7HZ(I,J)=)
      DO 4 I=1,K
4     WRITE(6,6) Z(I,1),Z(I,2)
6     FORMAT(1X,1HX,I2,2H--,1HY,I2)
      RETURN
      END
```

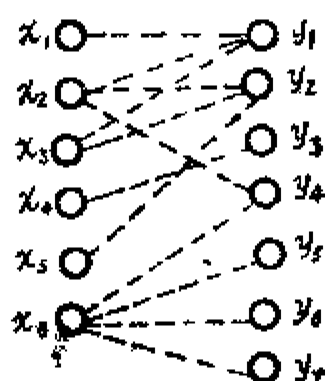


图 5.6

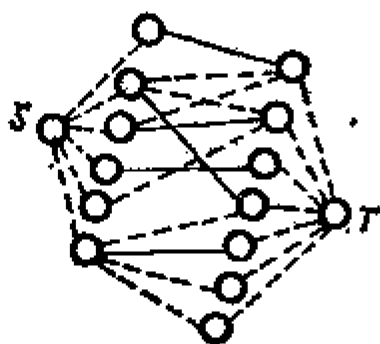


图 5.7

3）计算结果

X1——Y1

X2——Y4

X3 ——Y2

X4——Y3

X6——Y5

见图5.7。

5.3 一般图的最大基数匹配的算法

5.3.1 功　能

一般无向图的最大基数匹配的算法，其原理与二分图求

最大基数匹配的算法是一致的。但是由于一般图各顶点间的邻接关系比二分图复杂得多，自然它们的匹配关系也更加复杂。本节给出一个求一般图最大基数匹配的计算程序。

该算法在实践中是十分有用的。它不仅可以用于求二分图的最大基数匹配，更适合于处理非二分图情况下的二成员分组的问题。例如一批货物的分对问题，人员的分组问题等。

5.3.2 方法概述

关于计算一般图最大基数匹配的问题，1965年，Edmands提出了逐次调整的方法。该方法在一般图中生长M-交错树时，需要处理奇圈的情况，在计算机上实现起来较为复杂。如果我们在寻找M-交错树以扩大匹配时，用逐步对每一个顶点进行标号的方法可以避免对奇圈的处理。对于每一个未覆盖的顶点，生长一颗M-交错树，逐步扩大匹配。最终可以达到求出最大基数匹配的目的。

5.3.3 子程序参数说明

子程序名称 MAXGN(N,N1,A,D,K0)

N：图的顶点数目，整型变量。

N1：输出的匹配数组上界，N1=N/2，整型变量。

A(N,N)：图的顶点邻接矩阵的输入数组，整型数组。

其中：

$$A(i,j)=A(j,i)=\begin{cases}1, & \text{当第}i\text{顶点与第}j\text{顶点邻接时；}\\ 0, & \text{当第}i\text{顶点与第}j\text{顶点不邻接时。}\end{cases}$$

$i, j = 1, 2, \cdots, N$。

D(N1,2)：最大基数匹配计算结果的输出数据，整型数组。其中$D(i,1)$与$D(i,2)$为匹配的第i条弧的两端的顶点序号。

K0：最大基数匹配的基数输出数据，整型变量。

5.3.4 子 程 序

```
      SUBROUTINE MAXGN(N,N1,A,D,K0)
      INTEGER A(N,N),D(N1,2),U,V
      INTEGER S(30),S1(30),S3(30),S4(30),
   #  D0(30,2)
      K0=0
      DO 1 I=1,N
      S1(I)=0
      D0(I,1)=0
1     D0(I,2)=0
      DO 2 I=1,N1
      D(I,1)=0
      D(I,2)=0
      N0=N-1
      DO 4 I=1,N0
      J1=I+1
      DO 3 J=J1,N
      IF (A(I,J).NE.1.OR.S1(I)+S1(J)
   #  .NE.0) GOTO 3
      K0=K0+1
      D0(I,1)=I
      D0(I,2)=J
      D0(J,1)=J
```

```
      DO(J,2)=I
      S1(I)=1
      S1(J)=1
      GOTO 4
3     CONTINUE
4     CONTINUE
      DO 12 I=1,N
      IF (S1(I).EQ.1) GOTO 12
      DO 5 J=1,N
      S3(J)=0
      S4(J)=0
5     S(J)=0
      V=0
      DO 6 J=1,N
      IF A(I,J) .NE.1) GOTO 6
      V=V+1
      S3(J) =-11111
      S4(J)=1
6     CONTINUE
      IF (V.EQ.0) GOTO 12
13    DO 8 J=1,N
      IF (S4(J).LE.0) GOTO 8
      DO 7 K=1,N
      IF (A(K,J).EQ.1.AND.S3(K).EQ.0
    # .AND.DO(K,2).EQ.J) S3(K)=J
7     CONTINUE
8     CONTINUE
      V=0
      DO 10 K=1,N
      IF (S3(K).LE.0) GOTO 10
      DO 9 J=1,N
```

```
      IF(A(K,J).EQ.1.AND.S1(J).EQ.0)
    # GOTO 14
      IF (A(K,J).NE.1.OR.S4(J).NE.0.OR.
    # D0(K,2).EQ.J) GOTO 9
      V=V+1
      IF (S3(K).GT.0) S3(K)=-S3(K)
      S4(J)=K
9     CONTINUE
10    CONTINUE
      IF (V.GT.0) GOTO 13
      GOTO 12
14    DO 11 V=1,N
11    IF (S3(V).LT.0) S3(V)=-S3(V)
      S(1)=J
      S(2)=K
      U=3
      S1(J)=1
15    IF (S3(K).EQ.11111) GOTO 16
      J=S3(K)
      S(U)=J
      U=U+1
      K=S4(J)
      S(U)=K
      U=U+1
      GOTO 15
16    S1(K)=1
      KU=U-1
      DO 17 K0=2,KU,2
      K=KU+2-K0
      V=S(K)
      D0(V,1)=V
```

```
      D0(V,2)=S(K-1)
      V=S(K-1)
      D0(V,1)=V
17    D0(V,2)=S(K)
      K0=K0+1
12    CONTINUE
      V=0
      N0=N-1
      DO 18 I=1,N0
      I1=I+1
      DO 18 K=I1,N
      IF (D0(I,1).EQ.0.OR.D0(I,1).NE.
   #  D0(K,2)) GOTO 18
      V=V+1
      D(V,1)=D0(I,1)
      D0(K,1)=0
      D(V,2)=D0(I,2)
18    CONTINUE
      RETURN
      END
```

5.3.5 例　题

1）求以下十顶点图（见图5.8）的最大基数匹配。

$N=10$

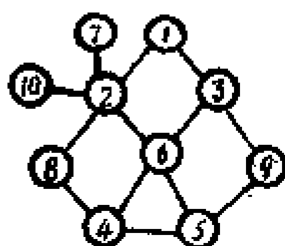


图　5.8

$$A(I,J)=\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2）计算程序

```
      INTEGER A(20,20),D(11,2)
      WRITE(7,1)
      READ(5,*) N
      N1=N/2
      CALL MAGN(N,N1,A,D)
1     FORMAT(1X,2HN=)
      STOP
      END
C
      SUBROUTINE MAGN(N,N1,A,D)
      INTEGER A(N,N),D(N1,2)
      WRITE(7,1)
1     FORMAT(1X,7HA(I,J)=)
      READ(5,2) ((A(I,J),J=1,I),I=1,N)
2     FORMAT(5I3)
      DO 44 I=1,N
      DO 44 J=1,I
44    A(J,I)=A(I,J)
```

```
      CALL MAXGN(N,N1,A,D,K0)
      WRITE(6,3)K0
3     FORMAT(1X,'KO=',I3)
      WRITE(6,4) (D(I,1),D(I,2),I=1,K0)
4     FORMAT(1X,'D=',15('(',I2,'--',I2,
   #  ')'))
      RETURN
      END
```

3）计算结果

K0=4

D=(1-- 3) (2--7) (4--8) (5 6)

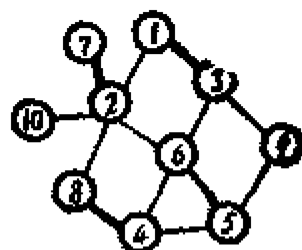


图 5.9

匹配的弧以粗线标在图5.9上。

图的极大权匹配与极大基数

5.4 匹配算法

5.4.1 功　能

前面介绍了求二分图与一般图的最大基数匹配的方法。由十分简单的例子都可以看出，相同基数的匹配结果不一定是唯一的。由不同的初始匹配出发，或者将原图的顶点顺序号改变，都有可能改变最大基数匹配的组合关系。所以，有时希望首先算出所有的极大基数匹配的方案，然后再极据特定的某些限制条件由这些匹配方案中选择几种来使用。

另外还有一类匹配的问题。如果某图的所有弧各有一个权值，怎样选择总权值达到极大值的匹配方案？称为“求图的极大权匹配”问题。

本节介绍一种算法，它既可以求一个图的多种不同的极

大基数匹配，又可以用来求极大权匹配。

首先，我们去求图上所有弧的全部极大独立集，从而得到该图的各种极大基数匹配或极大权匹配。

一般简单图的极大权匹配或极大基数匹配的算法有许多实用价值。例如考虑到效率的分配工作问题，将每个人对某种工作的效率作为权值，如何对一批人分配工作可以使总效率达到极大值？这就是一个求极大权匹配的问题。另外，考虑到利润达到极大值的生产计划问题等等，都可以作为求图的极大权匹配问题来处理。二分图的极大基数或极大权匹配问题，也可以用本节给出的方法来处理。

本节程序得到的结果较多，提供了从多种方案中优选的机会。但是也增加了计算量与计算机的存贮量。

5.4.2 方法概述

简单图G (V, E) 由N个顶点与M条弧构成。有时各弧还带有大小不等的权值。求图的极大基数匹配，要求匹配方案包含的弧数量大到极大值；而极大权匹配，要求匹配中全部弧的总权值在各种匹配方案中取极大值。

如果对原图$G(V,E)$的弧集E进行研究，根据各弧之间是否有公共顶点看作各弧是否邻接，将每条弧当作一个“顶点”看待。两条弧有一个共同的顶点，则认为它们之间有一条“弧”，这样来构成一个新的图G'。G'有M个“顶点”与M_s条“弧”。这里 $M_s=\sum_{\substack{i=1\\ s_i>1}}^{N} C_{s_i}^2$。其中，$C_r^2$表示$r$中取2的组合；$s_i$表示图$G$的$N$个顶点中的第$i$个与$M$条弧的关联次数。

求图G'的全部极大独立集。然后根据图G上各弧的权值，就能很容易地求出按照总权值降序排列的各种权匹配方案。

极大基数匹配的问题，可以当作所有弧的权值均为1的权匹配问题来使用本节的算法进行计算。

本算法调用了求图的全部极大独立集的布尔代数方法。这一算法介绍，可参考本书的有关段，这里不再重复。

本节的算法由两个子程序组成。其中，子程序GMA仅完成图G数据的输入，并根据特征参数KKK是否等于1决定是否输入权值。然后统计图G的N个顶点与M条弧的关联次数s_i，$i=1,2,\cdots,N$。求出图G'的总“弧”数

$$MK=\sum_{\substack{i=1\\ s_i>1}}^{N} C_{s_i}^{2}。$$

调用子程序GMAXGG，求极大权匹配的结果。

子程序GMAXGG完成以下几项任务：

1）形成图G'的关联矩阵。

2）调用求图的全部极大独立集的布尔代数算法子程序GMAXGL，求G'的全部极大独立集。

3）统计每个极大独立集的总权值，并按由大到小的降序重新排列顺序，输出结果。

5.4.3 子程序参数说明

1）子程序名称　GMA(N,M,A,LL,KK,P)

N：图G的顶点数目，整型变量。

M：图G的弧的数目，整型变量

A(N,N)：图G的邻接矩阵，进入子程序之后直接输入数据，整型数组。其中：

$A(i,j)=\begin{cases}1, & \text{图的第}i\text{顶点与第}j\text{顶点邻接时;}\\0, & \text{图的第}i\text{顶点与第}j\text{顶点不邻接时。}\end{cases}$

$i,j=1,2,\cdots,N$。

LL(40,N,2)：图 G 极大权匹配的输出数据，整型数组。在子程序中对该数据进行表格打印输出。其中$LL(i,j,1)$与$LL(i,j,2)$为第i个极大权匹配的第j条弧的两个顶点序号。极大权匹配方案总数为k_0时，有$LL(i,j,r)=0,i>k_0;j=1,2,\cdots,N;$ $r=1,2$。

KK(40)：全部极大权匹配包含的弧数目的输出数据，整型数组。各分量与数组LL对应。当$i>k_0$时，$KK(i)=0$，$i=k_0+1,k_0+2,\cdots,40$。

P(40)：全部极大权匹配的总权值的输出数据，整型数组。各分量与数组LL对应。当$i>k_0$时，$P(i)=0$ $i=k_0+1,k_0+2,\cdots,40$。

子程序局部量KKK：在子程序内直接输入一个整数，当输入值$KKK=1$时，求极大基数匹配。当输入值$KKK\neq 1$时，求极大权匹配，接着输入M条弧的权值。

2）子程序名称

GMAXGG(N,M,MK,G,V,A,P,LL,k_0,KK)

N：图G的顶点数目，整型变量。

M：图G的弧数目，整型变量。

MK：图G'的弧数目，输入数据，整型变量。

G(40,M)：求G'的极大独立集的工作数组，整型数组。

V(MK,2)：求G'的极大独立集的工作数组，整型数

组。

A(N,N)：图 G 的邻接矩阵，输入数据，整型数组。

P(40)：作为输入数据，前 M 个分量存放图 G 的 M 条弧的权值。计算结束后，作为输出数据，前 k_0 个分量存放 k_0 个极大权匹配各自的总权值，其余分量为零。整型数组。

LL(40,N,2)：图 G 的 k_0 个极大权匹配的计算结果。整型数组。其中 $LL(i,j,1)$，与 $LL(i,j,2)$ 为第 i 个极大权匹配的第 j 条弧的两个顶点的序号。当极大权匹配的总数为 k_0 时，有 $LL(i,j,r)\equiv 0$。$i>k_0$； $j=1,2,\cdots,N$； $r=1,2$。

KK(40)：图 G 的 k_0 个极大权匹配所含弧的数目的输出结果。整型数组。其中：

$$KK(i)\equiv 0，当 i=k_0+1，k_0+2,\cdots,40。$$

K0：图 G 的极大权匹配总数的输出结果，整型变量。

5.4.4 子程序

```
      SUBROUTINE GMA(N,M,A,LL,KK,P)
      INTEGER A(N,N),LL(40,N,2),KK(40),
     # G(40,30),CN(10),V(40,2),P(40)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=')
      DO 2 I=2,N
      I0=I-1
      READ(5,8) (A(I,J),J=1,I0)
8     FORMAT(30I1)
      DO 33 J=1,I0
```

```
83      A(J,I)=A(I,J)
2       CONTINUE
        DO 3 I=1,N
3       A(I,I)=1
        WRITE(7,20)
20      FORMAT(1X,'KKK=')
        READ(5,*)KKK
        IF (KKK.NE.1)GOTO 30
        DO 22 I=1,M
22      P(I)=1
        GOTO 40
30      WRITE(7,4)
4       FORMAT(/,1X,'P(I)=')
        READ(5,*) (P(I),I=1,M)
40      WRITE(6,7)N,M
7       FORMAT(1X,'N=',I2,' M=',I2,/)
        WRITE(6,1)
        DO 5 I=1,N
5       WRITE(6,6) (A(I,J),J=1,N)
6       FORMAT(1X,30I3)
        WRITE(6,4)
        WRITE(6,88) (P(I),I=1,M)
88      FORMAT(3(1X,15I3/))
        CN(1)=1
        DO 12 I=2,10
12      CN(I)=I*(I-1)/2
        MK=0
        DO 14 I=1,N
        MM=0
        DO 13 J=1,N
        IF (I.NE.J.AND.A(I,J).EQ.1)MM=MM+1
```

```
13      CONTINUE
        IF(MM.GT.1) MK=MK+CN(MM)
14      CONTINUE
        CALL GMAXGG(N,M,MK,G,V,A,P,LL,K0,
     #  KK)
        WRITE(6,9)K0
9       FORMAT(1X,'K0=',I3)
        DO 10 I=1,K0
        K1=KK(I)
10      WRITE(6,11)I,P(I),(LL(I,J,1),LL(I,
     #  J,2),J=1,K1)
11      FORMAT(1X,'I=',I2,' P(I)=',I3,3X,
     #  20(I2,'--',I2,2X))
        RETURN
        END
C
        SUBROUTINE GMAXGG(N,M,MK,G,V,A,P,
     #  LL,K0,KK)
        INTEGER A(N,N),P(40),LL(40,N,2),
     #  KK(40),G(40,M),L(30,2),V(MK,2),
     #  S(40),K0,S0
        K=0
        N0=N-1
        DO 11 I=1,N0
        I1=I+1
        DO 1 J=I1,N
        IF (A(I,J).EQ.0)GOTO 1
        K=K+1
        L(K,1)=I
        L(K,2)=J
1       CONTINUE
```

```
11      CONTINUE
        K=0
        K0=M-1
        DO 3 I=1,K0
        I1=I+1
        L1=L(I,1)
        L2=L(I,2)
        DO 2 J=I1,M
        L3=L(J,1)
        L4=L(J,2)
        IF (L1.NE.L3.AND.L1.NE.L4.AND.L2
     #  .NE.L3.AND.L2.NE.L4) GOTO 2
        K=K+1
        V(K,1)=I
        V(K,2)=J
2       CONTINUE
3       CONTINUE
        MK=K
        DO 4 I=1,40
        S(I)=0
        DO 4 J=1,M
4       G(I,J)=0
        CALL GMAXGL(MK,M,V,G,K0)
        DO 20 I=1,40
20      KK(I)=0
        DO 5 I=1,K0
        S0=0
        K1=0
        DO 6 J=1,M
        IF (G(I,J).EQ.0) GOTO 6
        S0=S0+P(J)
```

```
      K1=K1+1
6     CONTINUE
      S(I)=S0
5     KK(I)=K1
      DO 7 I=1,40
      DO 7 J=1,N
      LL(I,J,1)=0
7     LL(I,J,2)=0
      M0=K0-1
      DO 8 I=1,M0
      I1=I+1
      DO 9 J=I1,K0
      IF (S(I).GE.S(J)) GOTO 9
      II=S(I)
      S(I)=S(J)
      S(J)=II
      II=KK(I)
      KK(I)=KK(J)
      KK(J)=II
      DO 10 JJ=1,M
      II=G(I,JJ)
      G(I,JJ)=G(J,JJ)
10    G(J,JJ)=II
9     CONTINUE
8     CONTINUE
      K1=0
      DO 12 I=1, K0
      P(I)=0
      IF (KK(I).EQ.0) GOTO 12
      K=0
      K1=K1+1
```

```
      DO 13 J=1,M
      IF(G(I,J). EQ. 0) GOTQ 13
      K=K+1
      LL(K1,K,1)=L(J,1)
      LL(K1,K,2)=L(J,2)
13    CONTINUE
      P(K1)=S(K1)
12    CONTINUE
      RETURN
      END
```

5.4.5 例　题

1）用本节的算法解决下面两个问题。

（1）求十顶点二分图（见图5.10）的全部极大基数匹配和最大权匹配。

① 为了求全部极大基数匹配，令所有弧的权值$P(I)=1$。

$N=10, M=12$

$$A(I,J)=\begin{pmatrix} 1&0&0&0&0&1&1&1&0&0\\ 0&1&0&0&0&1&0&0&1&1\\ 0&0&1&0&0&0&1&0&1&0\\ 0&0&0&1&0&0&0&1&0&0\\ 0&0&0&0&1&1&0&1&0&1\\ 1&1&0&0&1&1&0&0&0&0\\ 1&0&1&0&0&0&1&0&0&0\\ 1&0&0&1&1&0&0&1&0&0\\ 0&1&1&0&0&0&0&0&1&0\\ 0&1&0&0&1&0&0&0&0&1 \end{pmatrix}$$

P(I)＝(1 1 1 1 1 1 1 1 1 1 1 1)

② 为了求图5.9的全部极大权匹配，各弧的权值标在图5.11上面。

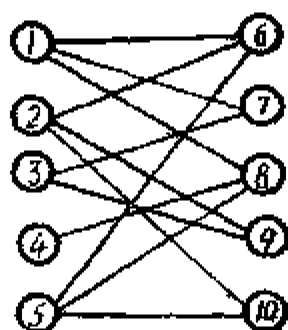


图 5.10

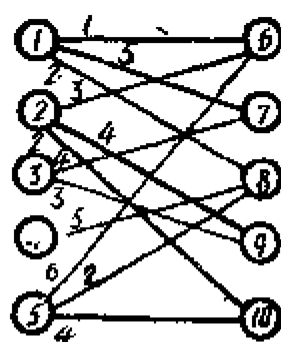


图 6.11

$N=10, M=12$

$A(I,J)$与①相同。

$P(I)$＝(1 5 2 3 4 2 4 3 5 6 2 4)

（2）乒乓球队拟从九名男队员中尽可能多地组成男子双打组。考虑到这些队员之间相互配合效果存在的差异，并考虑到每个队员都不允许编入两个双打组而同时出场的前题条件下，求一个使全队出场后总积分值最理想的编组方案。此题可以利用本节的算法来解决。为此作图5.12，将队员编号作为图的顶点号；将事先测得的队员间互相配合效果的统计值当作弧的权值。

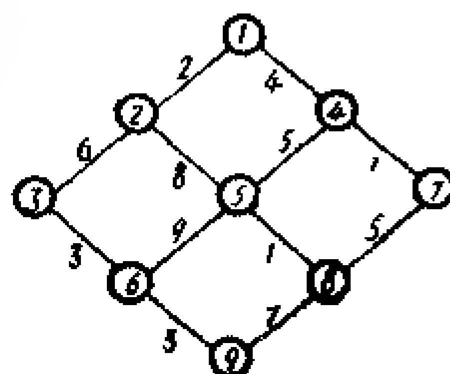


图 5.12

$N=9$，$M=12$

$$A(I,J)=\begin{pmatrix} 1&1&0&1&0&0&0&0&0\\ 1&1&1&0&1&0&0&0&0\\ 0&1&1&0&0&1&0&0&0\\ 1&0&0&1&1&0&1&0&0\\ 0&1&0&1&1&1&0&1&0\\ 0&0&1&0&1&1&0&0&1\\ 0&0&0&1&0&0&1&1&0\\ 0&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&1&0&1&1 \end{pmatrix}$$

$P(I)=(2\ 4\ 6\ 8\ 3\ 5\ 7\ 9\ 1\ 3\ 5\ 7)$

2）计算程序

```
      INTEGER  A(20,20),LL(40,20,2),P(40)
#     ,KK(40)
      WRITE(7,1)
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)M
      CALL GMA(N,M,A,LL,KK,P)
1     FORMAT(1X,'N=')
2     FORMAT(1X,'M=')
      STOP
      END
```

3）计算结果

（1）全都极大基数匹配和极大权匹配分别见①和②。

① $K_0=24$

总权值		编	组			
1	5	1--7	2--6	3- 9	4 - 8	5--10
2	5	1 6	2 -9	3--7	4 8	5--10

3	5	1--7	2--10	3--9	4--8	5 6
4	4	1--8	2--10	3--7	5--6	
5	4	2--6	3--7	4--8	5--10	
6	4	1--6	3--9	4--8	5--10	
7	4	1--6	2--10	3 9	5--8	
8	4	1--6	2--10	3--9	4--8	
9	4	1--6	2--10	3--7	5--8	
10	4	1--6	2--10	3--7	4--8	
11	4	2--10	3--7	3--8	5--6	
12	4	1--7	2--10	3--9	5--8	
13	4	1--8	2--9	3--7	5--10	
14	4	1--8	2--9	3--7	5--6	
15	4	1--7	2--9	4 8	5--10	
16	4	1--7	2--9	4--8	5--6	
17	4	2--9	3--7	4--8	5--6	
18	4	1--7	2--6	3--9	5--8	
19	4	1--6	2--9	3--7	5--8	
20	4	1--8	2--6	3--9	5--10	
21	4	1--8	2--6	3--7	5--10	
22	4	1--8	2--10	3 9	5--6	
23	3	2--6	3--7	5--8		
24	3	1--7	2--9	5--8		

② $K_0=24$

	总权值	编	组			
1	21	1--7	2--10	3--9	4--8	5--6
2	20	1--7	2--6	3--9	4--8	5--10
3	20	1--7	2--9	4 8	5--6	
4	19	2--9	3--7	4--8	5--6	
5	18	1--6	2--9	3 7	4--8	5--10
6	18	1--7	2--9	4--8	5--10	
7	17	2--10	3--7	4--8	5--6	

8	16	2--6	3--7	4--8	5--10
9	16	1--8	2--9	3--7	5--6
10	14	1--8	2--9	3--7	5--10
11	14	1--8	2--10	3--7	5--6
12	13	1--7	2--6	3--9	5--8
13	13	1--8	2--6	3--7	5--10
14	13	1--8	2--10	3--9	5--6
15	13	1--6	3--9	4--8	5--10
16	12	1--6	2--10	3--7	4--8
17	12	1--8	2--6	3--9	5--10
18	12	1--7	2--10	3--9	5--8
19	11	1--6	2--10	3--9	4--8
20	11	1--7	2--9	5--8	
21	11	1--6	2--9	3--7	5--8
22	9	1--6	2--10	3--7	5--8
23	9	2--6	3--7	5--8	
24	8	1--6	2--10	3--9	5--8

③ $K_0=22$

	总权值	编组			
1	29	2--3	4--7	5--6	8--9
2	26	1--4	2--3	5--6	8--9
3	25	1--2	4--7	5--6	8--9
4	25	2--5	3--6	4--7	8--9
5	24	1--4	2--3	5--6	7--8
6	22	1--4	2--5	3--6	8--9
7	20	1--4	2--5	6--9	7--8
8	20	1--4	2--5	3--6	7--8
9	19	1--2	3--6	4--7	8--9
10	19	2--3	4--5	6--9	7--8
11	18	2--3	4--5	8--9	
12	18	1--4	2--3	6--9	7--8

13	18	2 -- 5	4 -- 7	6 9	
14	17	2 -- 3	4 -- 7	5 -- 8	6 -- 9
15	17	1 - 2	3 -- 6	4 -- 5	8 - 9
16	16	1 - 2	5 -- 6	7 - 8	
17	15	1 - 2	4 - 5	6 9	7 -- 8
18	15	1 - 2	3 -- 6	4 -- 5	7 -- 8
19	14	1 -- 4	2 -- 3	5 8	6 -- 9
20	13	1 2	4 -- 7	5 8	6 -- 9
21	13	1 - 2	3 -- 6	4 - 7	5 -- 8
22	8	1 4	3 -- 6	5 8	

6. 着色问题的计算

6.1 图的顶点着色的纵深搜索法

6.1.1 功　能

图G(V,E)由N个顶点和M条弧构成。如果对该图的顶点进行着色，要求相邻接的顶点着不同的颜色，至少需要几种不同的颜色？各个顶点的颜色应当如何分配？这就是所谓图的顶点着色问题。

许多实用问题可以化为着色问题来处理。另外，图的弧着色问题也可以化为顶点着色的问题来处理。

本节介绍一种算法，在确定了色数之后，利用纵深搜索法寻找多种不同的着色方案。

6.1.2 方法概述

为了对图G（V,E）进行顶点着色，首先对这些顶点进行编号，给出顶点的邻接矩阵。如果要求最多用K0种色数对顶点着色，希望得KK种不同的着色方案。在此使用了图论中常用的一种DFS纵深搜索法。具体步骤按流程图6.1进行。

在图6.1中，出口1为该图若按K0种色数着色，不存在KK种不同的着色方案的非正常出口。出口2是找到KK种

着色方案后的正常出口。

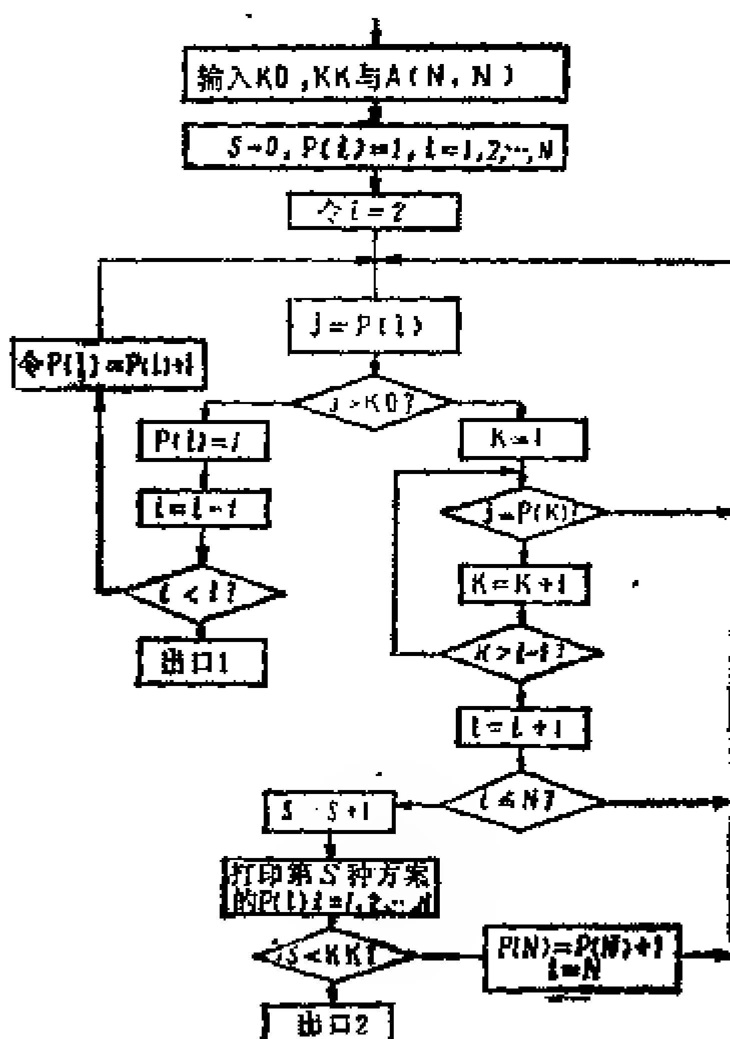


图 6.1

6.1.3 子程序参数说明

子程序名称　DFSASY (N,A,P,K0,KK)

N：图的顶点数目，整型变量。

A(N,N)：图的顶点邻接矩阵的输入数据，整型数组。其中：

$$A(i,j)=\begin{cases}1, & \text{第}i\text{顶点与第}j\text{顶点邻接；}\\0, & \text{第}i\text{顶点与第}j\text{顶点不邻接。}\end{cases}$$

$i,j=1,2,\cdots,N$。

P(N)：工作单元，整型数组。如果图存在K0种色数的着色方案，P (N) 在计算结束时存放一种着色方案。其中：P(i)=K，表示第i顶点着第K种颜色，$1 \leqslant K \leqslant K0$；i=1，2…，N。

注意：图的K0色数的KK种着色方案将在子程序内输出。

K0：色数的输入数据，整型变量。

KK：方案数的输入数据，整型变量。

6.1.4 子 程 序

```
      SUBROUTINE DFSASY(N,A,P,K0,KK)
      INTEGER A(N,N),P(N),S
      S=0
      DO 2 I=1,N
2     P(I)=1
      I=2
3     J=P(I)
      IF (J.LT.K0+1)GOTO 4
      P(I)=1
      I=I-1
      IF(I.LT.1) GOTO 12
      P(I)=P(I)+1
      GOTO 3
4     I0=I-1
      IF(I0.LT.1) GOTO 6
      DO 5 K=1,I0
      IF(P(K)*A(I,K).NE.J)GOTO 5
      P(I)=P(I)+1
      GOTO 3
```

```
5       CONTINUE
6       CONTINUE
        I=I+1
        IF(I.LE.N)GOTO 3
        S=S+1
        WRITE(6,7)S,(P(K),K=1,N)
7       FORMAT(1X,'S=',I2,2X,30I2)
        IF (S.GE.KK)GOTO 12
        P(N)=P(N)+1
        I=N
        GOTO 3
12      CONTINUE
        RETURN
        END
```

6.1.5 例　题

1）对以下三图进行顶点着色。

（1）对六顶点图（见图6.2）进行顶点着色，要求色数为 4，给出十种不同的着色方案。

N＝6，K0＝4，KK＝10

$$A(I,J)=\begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{vmatrix}$$

（2）对N＝11的图（见图6.3），作K0＝4，KK＝10的顶点着色。

$$A(I,J)=\begin{vmatrix}1&1&1&1&1&1&0&0&0&0&0\\1&1&1&1&0&0&0&0&0&0&0\\1&1&1&1&0&0&0&0&1&1&0\\1&1&1&1&1&1&1&1&1&1&1\\1&0&0&1&1&1&0&0&0&0&0\\1&0&0&1&1&1&1&1&0&0&0\\0&0&0&1&0&1&1&1&0&0&0\\0&0&0&1&0&1&1&1&0&1&1\\0&0&1&1&0&0&0&0&1&1&0\\0&0&1&1&0&0&0&1&1&1&1\\0&0&0&1&0&0&0&1&0&1&1\end{vmatrix}$$

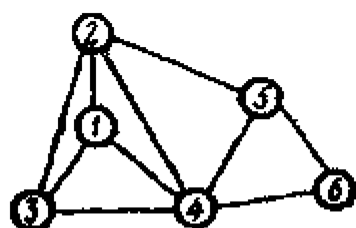


图 6.2

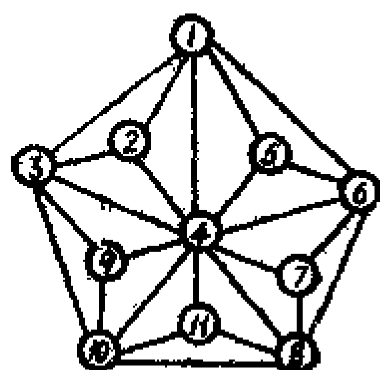


图 6.3

（3）对我国地图进行着色。各地区的编号如下：

$N=32$

$K0=4$，$KK=10$

$A(I,J)=$

```
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0
0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1
```

2）计算程序

```
INTEGER A(40,40),P(40)
WRITE(7,1)
```

```
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)K0
      WRITE(7,3)
      READ(5,*)KK
      CALL SAS(N,A,P,K0,KK)
1     FORMAT(1X,2HN=)
2     FORMAT(1X,3HK0=)
3     FORMAT(1X,3HKK=)
      STOP
      END
```

黑龙江	吉林	辽宁	内蒙	河北	北京	天津	山西	山东	江苏	安徽	上海	浙江	江西	福建	台湾
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
河南	湖北	湖南	广东	广西	陕西	宁夏	甘肃	青海	新疆	四川	贵州	云南	西藏	香港	澳门
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

```
C
      SUBROUTINE SAS(N,A,P,K0,KK)
      INTEGER A(N,N),P(N)
      WRITE(7,2)
      DO 3 I=2,N
      I1=I-1
3     READ(5,10) (A(I,J),J=1,I1)
10    FORMAT(40I1)
      WRITE(6,1)K0
1     FORMAT(1X,'K0=',I2)
      DO 6 I=1,N
```

```
      A(I,I)=1
      DO 6 J=I,N
6     A(I,J)=A(J,I)
      CALL DFSASY(N,A,P,K0,KK)
      RETURN
      END
```

3）计算结果

（1）

```
S=1   1 2 3 4 1 2
S=2   1 2 3 4 1 3
S=3   1 2 3 4 3 1
S=4   1 2 3 4 3 2
S=5   1 2 4 3 1 2
S=6   1 2 4 3 1 4
S=7   1 2 4 3 4 1
S=8   1 2 4 3 4 2
S=9   1 3 2 4 1 2
S=10  1 3 2 4 1 3
```

（2）

```
S=1   1 2 3 4 2 3 1 2 2 1 3
S=2   1 2 3 4 2 3 2 1 1 2 3
S=3   1 2 3 4 3 2 1 3 1 2 1
S=4   1 2 3 4 3 2 1 3 2 1 2
S=5   1 2 3 4 3 2 3 1 1 2 3
S=6   1 2 4 3 2 4 1 2 2 1 4
S=7   1 2 4 3 2 4 2 1 1 2 4
S=8   1 2 4 3 4 2 1 4 1 2 1
S=9   1 2 4 3 4 2 1 4 2 1 2
S=10  1 2 4 3 4 2 4 1 1 2 4
```

（3）

```
S=1  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 1 2
S=2  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 1 3
S=3  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 2 1
S=4  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 2 3
S=5  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 3 1
S=6  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 3 4 2 3 1 3 2
S=7  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 4 4 2 3 1 1 2
S=8  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 4 4 2 3 1 1 3
S=9  1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 4 4 2 3 1 2 1
S=10 1 2 1 3 2 1 3 1 1 2 3 1 4 2 1 2 4 1 3 4 1 2 4 1 2 4 4 2 3 1 2 3
```

6.2 图的顶点色数及着色方案的求法

6.2.1 功　能

对图G（V，E）的顶点着色问题，经常需要研究这个图的顶点至少需要多少种不同的颜色，这就是求顶点的色数的问题。

求色数及具体给出着色方案一类问题，有广泛的实用价值。例如，地图上各国至少需要着几种不同的颜色，排课程表的问题，排时间表的问题；物资的分类贮存问题，都能够用到求图的顶点色数，并用求着色方案的方法来处理这些问题。本节的程序仍然使用了DFS方法。

6.2.2 方法概述

对N顶点的图G(V,E)，给定邻接矩阵A(N,N)之后，由色数K0=1开始进行搜索。如果存在一种K0色数的着色方案，则K0就是该图顶点的色数，否则，令K0=K0+1，重新进行搜索，直到找到一种顶点着色方案为止。每一步的

搜索方法都按照DFS纵深搜索法。

本节程序与上一节不同的是，求出色数后，只给出一种着色方案便结束。

6.2.3 子程序参数说明

子程序名称 DFSASS(N,A,P,K0)

N：图的顶点数目，整型变量。

A(N,N)：图的邻接矩阵的输入数据，整型数组。

其中，

$$A(i,j)=\begin{cases}1,\text{当第}i\text{顶点与第}j\text{顶点相邻接;}\\0,\text{当第}i\text{顶点与第}j\text{顶点不邻接。}\end{cases}$$

$i,\ j=1,2,\cdots,\ N$。

P(N)：着色方案的输出结果，整型数组。其中 P(i)=K　i=1，2，…N；$1\leqslant K\leqslant K0$。K0是该图的色数。

K0：色数的计算结果，整型变量。标明该图着色至少要用K0种不同的颜色。

6.2.4 子 程 序

```
      SUBROUTINE DFSASS(N,A,P,K0)
      INTEGER A(N,N),P(N)
      K0= 1
      DO 2 I=1,N
2     P(I)= 1
1     I= 2
3     J= P(I)
      IF(J.LT.K0+ 1) GOTO 4
```

```
      P(I)=1
      I=I-1
      IF (I.GE.1) GOTO 8
      K0=K0+1
      GOTO 1
8     P(I)=P(I)+1
      GOTO 3
4     I0=I-1
      IF (I0.LT.1) GOTO 6
      DO 5 K=1,I0
      IF(P(K)*A(I,K).NE.J) GOTO 5
      P(I)=P(I)+1
      GOTO 3
5     CONTINUE
6     CONTINUE
      I=I+1
      IF (I.LE.N) GOTO 8
      WRITE(6,7) (P(K),K=1,N)
7     FORMAT(1X,'P(K)=',30I2)
      RETURN
      END
```

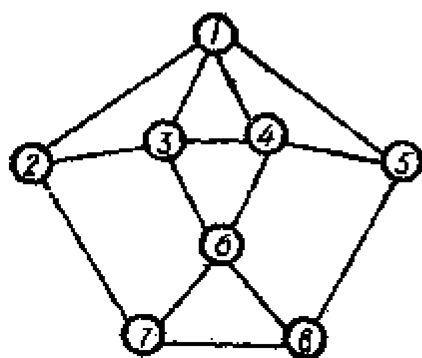


图 6.4

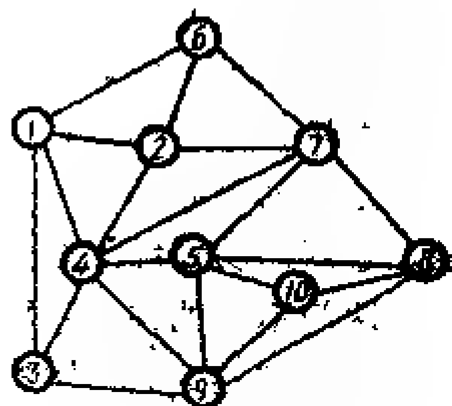


图 6.5

6.2.5 例　题

1）求以下二图的顶点色数，并分别给出着色方案.

（1）见图6.4。

$N=8$

$$A(I,J)=\begin{vmatrix} 1&1&1&1&1&0&0&0\\ 1&1&1&0&0&0&1&0\\ 1&1&1&1&0&1&0&0\\ 1&0&1&1&1&1&0&0\\ 1&0&0&1&1&0&0&1\\ 0&0&1&1&0&1&1&1\\ 0&1&0&0&0&1&1&1\\ 0&0&0&0&1&1&1&1 \end{vmatrix}$$

（2）对某班学生的十门选修课程进行考试。由于同学们选修课程互有重复，为了使全班学生都能够参加所选修课程的考试，问至少应当安排几场不同时间的考试？哪些课程可以安排在间时考试？

这一问题归结为求十个顶点的图的色数及着色方案问题。这10门功课经过编号作为图的顶点。只要同一个学生选修了两门不同的课，这两个顶点间就存在一条弧。最终这十门功课的邻接关系如图6.5所示。

$N=10$

$$A(I,J)=\begin{vmatrix} 1&1&1&1&0&1&0&0&0&0\\ 1&1&0&1&0&1&1&0&0&0\\ 1&0&1&1&0&0&0&0&1&0\\ 1&1&1&1&1&0&1&0&1&0\\ 0&0&0&1&1&0&1&1&1&1\\ 1&1&0&0&0&1&1&0&0&0\\ 0&1&0&1&1&1&1&1&0&0\\ 0&0&0&0&1&0&1&1&1&1\\ 0&0&1&1&1&0&0&1&1&1\\ 0&0&0&0&1&0&0&1&1&1 \end{vmatrix}$$

2）计算程序

```
      INTEGER A(30,30),P(30)
      WRITE(7,1)
      READ(5,*)N
      CALL SSS(N,A,P)
1     FORMAT(1X,2HN=)
      STOP
C     END
      SUBROUTINE SSS(N,A,P)
      INTEGER A(N,N),P(N)
      WRITE(7,2)
      DO 3 I=2,N
      I1=I-1
3     READ(5,10) (A(I,J),J=1,I1)
10    FORMAT(50I1)
      WRITE(6,8)
8     FORMAT(1X,'A(I,J)=',1)
      DO 6 I=1,N
      A(I,I)=1
```

```
      DO 6 J=I,N
6     A(I,J)=A(J,I)
      DO 4 I=1,N
      WRITE(6,7) I,(A(I,J),J=1,N)
4     CONTINUE
2     FORMAT(1X,7HA(I,J)=)
7     FORMAT(1X,I3,3X,30I2)
      WRITE(6,9)
9     FORMAT(/)
      CALL DFSASS (N,A,P,K0)
      WRITE(6,1) K0
1     FORMAT(1X,'K0= ,I2)
      RETURN
      END
```

3）计算结果

(1) K0=3

P(K)=(1 2 3 2 3 1 3 2)

图6.4至少需要用三种不同的颜色进行顶点着色。由上面的着色方案，各顶点根据着色相同可分为以下三组：

(1，6)，(2，4，8)，(3，5，7)。

(2) K0=4

P(K)=(1 2 2 3 1 3 4 2 4 3)

由计算结果表明，对该班至少要安排四次不同时间的考试。可以安排在同一时间内进行考试的课程分别是以下四组：(1，5) (2，3，8) (4，6，10) (7，9)。

6.3 图的弧色数以及着色方案的算法

6.3.1 功　能

上一节介绍了图的顶点着色的算法。与图的顶点着色类似的是图的弧着色问题。连通图G(V,E)由N个顶点与M条弧构成。当两条弧有一个公共的顶点时，认为这两条弧相邻。对图上的每一条弧进行着色，要求有公共顶点的弧着不同的颜色，问至少需要几种不同的颜色？这是弧的色数问题。本节的程序给出求图的弧色数的方法，同时得出弧着色的方案。

6.3.2 方法概述

已知一个N顶点M条弧的连通图G(V,E)。图的各顶点之间的邻接矩阵是容易写出的，我们就从输入该图的顶点邻接矩阵开始。

$$A(i,j)=\begin{cases}1\text{，当第i顶点与第j顶点邻接，}\\0\text{，当第i顶点与第j顶点不邻接。}\end{cases}$$

$$i,j=1,2,\cdots N。$$

然后，由A(N,N)的数据可以直接计算出图上各弧之间的相邻关系矩阵B(M,M)。其中：

$$B(i,j)=\begin{cases}1\text{，当第i弧与第j弧有公共顶点，}\\0\text{，当第i弧与第j弧无公共顶点。}\end{cases}$$

$$i,j=1,2,\cdots,M。$$

对于矩阵B(M,M)，使用纵深搜索法就可以求出色数，并求出弧的着色方案。在本节的计算结果中，弧着色方案的输

出，不给出各弧的编号，而按照每条弧的两端顶点序号给出着色编号。

6.3.3 子程序参数说明

子程序名称DFSAAA(N,M,A,P,B,Q,K0)

N：图的顶点数目，整型变量。

M：图的弧的数目，整型变量。

A(N,N)：图上各顶点的邻接矩阵，输入数据，整型数组。其中，

$$A(i,j)=\begin{cases}1\ \text{，第i顶点与第j顶点邻接时；}\\0\ \text{，第i顶点与第j顶点不邻接时。}\end{cases}$$

$i,j=1,2,\cdots,N$。

P(M,3)：图的弧着色的一组输出结果，整型数组。其中P(i,1)与P(i,2)分别存放第i条弧的两端顶点的序号，P(i,3)为这条弧应当着色的序号，$i=1,2,\cdots,M$。

B(M,M)：中间工作单元，整型数组。

Q(M)：中间工作单元，整型数组。

K0：图的弧着色的色数输出结果，整型变量。

注：本子程序调用了求顶点色数的子程序DFSASS。关于这个子程序的内容，请参考前面6.2段的介绍。

6.3.4 子　程　序

```
      SUBROUTINE DFSAAA(N,M,A,P,B,Q,K0)
      INTEGER A(N,N),P(M,3),B(M,M),Q(M),
     #  L(30,2)
      K=0
```

```
        N0=N-1
        DO 11 I=1,N0
        I1=I+1
        DO 1 J=I1,N
        IF(A(I,J).EQ.0)GOTO 1
        K=K+1
        L(K,1)=I
        L(K,2)=J
1       CONTINUE
11      CONTINUE
        K0=M-1
        DO 2 I=1,K0
        I1=I+1
        L1=L(I,1)
        L2=L(I,2)
        DO 2 J=I1,M
        L3=L(J,1)
        L4=L(J,2)
        IF(L1.NE.L3.AND.L1.NE.L4.AND.L2
     # .NE.L3.AND.L2.NE.L4)GOTO 3
        B(I,J)=1
        GOTO 2
3       B(I,J)=0
2       B(J,I)=B(I,J)
        DO 4 I=1,M
4       B(I,I)=1
        CALL DFSASS(M,B,Q,K0)
        DO 5 I=1,M
        P(I,1)=L(I,1)
        P(I,2)=L(I,2)
5       P(I,3)=Q(I)
```

```
RETURN
END
```

6.3.5 例　　题

1）求以下三个图的弧色数，并分别给出一种弧着色的方案。将图上各弧的色数顺序号标在图上弧的侧面。

（1）见图6.6。

$N=4$，$M=5$

$$A(I,J)=\begin{pmatrix}1&1&0&1\\1&1&1&1\\0&1&1&1\\1&1&1&1\end{pmatrix}$$

（2）见图6.7。

$N=9$，$M=12$

$$A(I,J)=\begin{pmatrix}1&1&0&1&0&0&0&0&0\\1&1&1&0&1&0&0&0&0\\0&1&1&0&0&1&0&0&0\\1&0&0&1&1&0&1&0&0\\0&1&0&1&1&1&0&1&0\\0&0&1&0&1&1&0&0&1\\0&0&0&1&0&0&1&1&0\\0&0&0&0&1&0&1&1&1\\0&0&0&0&0&1&0&1&1\end{pmatrix}$$

（3）见图6.8。

$N=11$，$M=13$

$$A(I,J)=\begin{pmatrix}1&1&1&0&0&0&0&0&1&1&0\\1&1&0&1&1&0&0&0&0&0&0\\1&0&1&0&1&1&0&0&0&0&0\\0&1&0&1&0&0&1&0&1&0&0\\0&1&1&0&1&0&1&1&0&0&0\\0&0&1&0&0&1&0&1&0&1&0\\0&0&0&1&1&0&1&0&0&0&1\\0&0&0&0&1&1&0&1&0&0&1\\1&0&0&1&0&0&0&0&1&0&1\\1&0&0&0&0&1&0&0&0&1&1\\0&0&0&0&0&0&1&1&1&1&1\end{pmatrix}$$

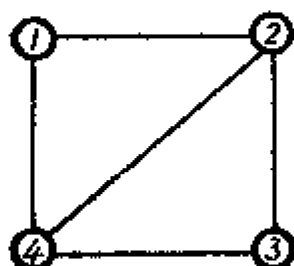


图 6.6

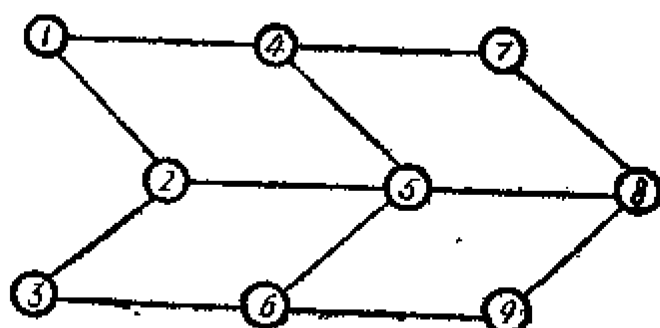


图 6.7

2）计算程序

```
      INTEGER A(30,30),P(40,3)
      WRITE(7,1)
      READ(5,*)N
      WRITE(7,2)
      READ(5,*)M
      CALL DFS(N,M,A,P)
1     FORMAT(1X,'N=')
2     FORMAT(1X,'M=')
      STOP
```

```
      END
C
      SUBROUTINE DFS(N,M,A,P)
      INTEGER A(N,N),P(M,3),B(40,40),
   #  Q(40)
      WRITE(7,1)
1     FORMAT(1X,'A(I,J)=')
      DO 2 I=2,N
      I0=I-1
2     READ(5,8)(A(I,J),J=1,I0)
8     FORMAT(30I1)
      DO 3 I=2,N
      I0=I-1
      DO 3 J=1,I0
3     A(J,I)=A(I,J)
      DO 7 I=1,N
7     A(I,I)=1
      WRITE(6,40)N,M
40    FORMAT(1X,'N=',I2,4X,'M=',I2,/)
      WRITE(6,1)
      DO 22 I=1,N
22    WRITE(6,33)(A(I,J),J=1,N)
33    FORMAT(1X,30I3)
      CALL DFSAAA(N,M,A,P,B,Q,K0)
      DO 6 I=1,M
6     WRITE(6,4)I,(P(I,J),J=1,3)
4     FORMAT(1X,'I=',I2,2X,I2,'--',I2,
   #  1X,'(',I2,')')
      WRITE(6,5)K0
5     FORMAT(1X,'K0=',I2)
      RETURN
```

END

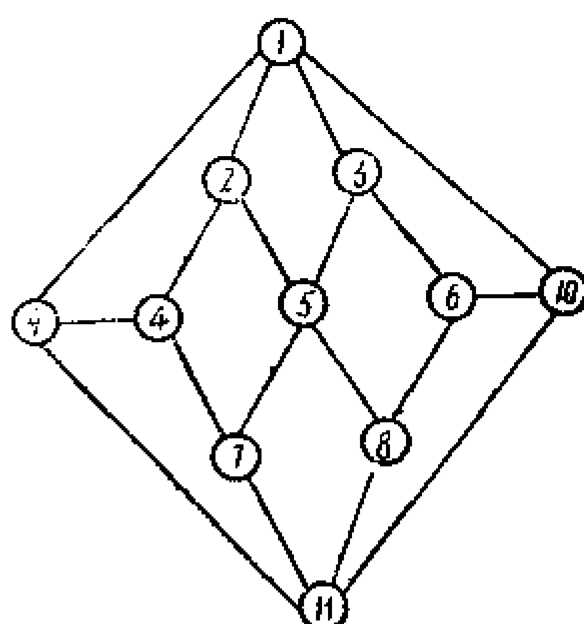


图 6.8

3）计算结果

（1）K0=3

I=1　1—2（1）

I=2　1—4（2）

I=3　2—3（2）

I=4　2—4（3）

I=5　3—4（1）

见图6.9。

（2）K0=4

I=1　1—2（1）

I=2　1—4（2）

I=3　2—3（2）

I=4　2—5（3）

I=5　3—6（1）

I=6　4—5（1）

I=7　4—7（3）

I = 8　5 — 6 (2)

I = 9　5 — 8 (4)

I = 10　6 — 9 (3)

I = 11　7 — 8 (1)

I = 12　8 — 9 (2)

见图6.10。

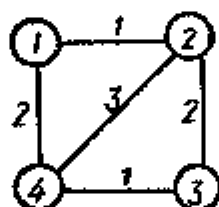


图 6.9

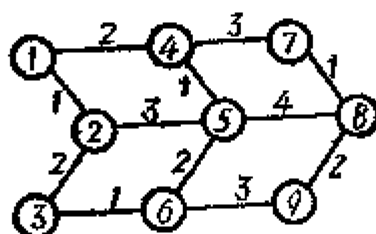


图 6.10

(3) K0 = 4

I = 1　1 — 2 (1)

I = 2　1 — 3 (2)

I = 3　1 — 9 (3)

I = 4　1 — 10 (4)

I = 5　2 — 4 (2)

I = 6　2 — 5 (3)

I = 7　3 — 5 (1)

I = 8　3 — 6 (3)

I = 9　4 — 7 (1)

I = 10　4 — 9 (4)

I = 11　5 — 7 (2)

I = 12　5 — 8 (4)

I = 13　6 — 8 (1)

I = 14　6 — 10 (2)

I = 15　7 — 11 (4)

I = 16　8 — 11 (2)

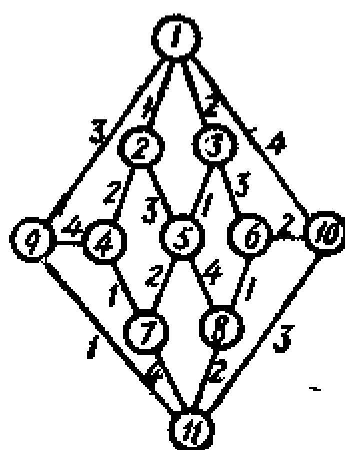


图 6.11

I = 17　9 — 11（1）

I = 18　10 — 11（8）

见图6.11。

以上各例的结果列出了每条弧的两个顶点的顺序号，括号内列出了该弧着色序号。

7. 网络流的算法

7.1 网络最大流的算法

7.1.1 功　能

在图论的应用中，寻找运输网络中固定的出发点至终点的最大流量运输方案的算法是十分有用的。这种算法就称为网络最大流的算法。在给定网络中每一条弧的最大允许流量之后，使用本节的子程序可以求出从图上一指定顶点作为出发点，至另一个指定的顶点作为终点的最大流量，以及这个最大流量运输方案分配到图上各弧的运输流量的数值。该算法对交通运输方案的制定、物资紧急调运等方面十分有用。

如果是多出发点与多终点的运输流问题，也可以用本节的程序来计算。设网络上这些出发点为S_i($i=1,2,\cdots,r$)，终点为T_j（$j=1,2,\cdots,v$）。在网络上增加两个虚设的顶点S_0与T_0。将S_0与全部出发点S_i用虚设的流量值为∞的弧L_i连接，作为新的单出发点。将T_0与全部终点T_j用虚设的流量值为∞的弧L_j连接，作为新的单终点。然后，用本节的子程序求S_0至T_0的最大流，也就得到了原问题的最大流运输方案。

7.1.2 方法流连

已知N个顶点运输网络G上各顶点间的最大允许流量矩

阵记为C。求该图上从出发点S到终点T的最大流量运输方案的总流量，以及这个运输方案分配到图上各弧的流量值。

设x与y为图上的两个顶点，用（x，y）表示由顶点x到顶点y的一条弧，以$f(x,y)$表示通过弧(x,y)的流量。弧(x,y)在流量矩阵中对应的允许流量为$C(x,y)$。如果$f(x,y)$为一个由出发点S至终点的可行流，必须满足下列条件：

$$0 \leqslant f(x,y) \leqslant C(x,y)$$

$$\sum_{y \leftarrow x} f(x,y) - \sum_{x \leftarrow y} f(y,x) = \begin{cases} 0, & \text{当 } x \neq S,\ T \text{ 时,} \\ V, & \text{当 } x = S, \\ -V, & \text{当 } x = T。\end{cases}$$

V就是这一可行流的流量。

在所有可行流当中找一个使V取极大值的流，称为最大流。就得到了本算法的结果。

本节的子程序是根据Ford与Fulkerson给出的算法编制的，具体步骤如下：

步骤1 设$f(x,y)$是任意一个可行流。为了处理的方便，程序的开头一律从零流开始。给出发点S一个标号$(-, M0)$。此处的$M0$为一个充分大的数，用以代替$+\infty$。

步骤2 标号与检查。

① 如果所有的标号都已经检查过了，仍然找不到新的标号点，转向步骤4。

② 如果找到一个已经标号但未检查的点x，则检查与x邻接但尚未标号的点y。每一条弧(x,y)，如果满足$f(x,y) < C(x,y)$，且y尚未标号，那么给y以标号$(+x, \delta(y))$。其中：

$$\delta(y) = \min\{C(x,y) - f(x,y), \delta(x)\}$$

每一条弧(y,x)，如果有$f(y,x) > 0$，且y尚未标号，那

么给 y 以标号$(-x,\delta(y))$。其中：

$$\delta(y)=\min\{f(y,x),\delta(x)\}$$

③ 如果终点T已经标号，则转步骤3；否则，继续执行步骤2。

步骤3 寻找增广路。由终点T出发，进行逆向追踪，找一条由S至T的增广路。在这条增广路上的每一条弧，根据其标号的正与负，增加或减少流量$\delta(T)$。然后返回步骤1。

步骤4 找到了由S至T的最大流，计算结束。

为了保证本算法的有效性，程序的执行过程中，对先标号者先检查，以较快的速度找到最大流的结果。

7.1.3 子程序参数说明

子程序名称 MAXC(N,C,L,S,T,M0,MC)

N：图的顶点数目，整型变量。

C(N,N)：图的允许流量输入矩阵，整型数组。

L(N,N)：最大流分配到每条弧上的流量的输出矩阵，整型数组。其中，$L(i,j)$存放第i顶点至第j顶点的弧上的分配流量。

S：出发点序号，整型变量。

T：终点序号，整型变量。

M0：流量极大值，输入一个超过图上所有弧的流量之和的整数，整型变量。

MC：图上S至T的最大流量的输出值，整型变量。

7.1.4 子程序

```
SUBROUTINE MAXC(N,C,L,S,T,M0,MC)
```

```
      INTEGER C(N,N),L(N,N),S,T
      INTEGER P(30),Q(30),R(30),D(30,3)
      MC=0
      DO 7 I=1,N
      DO 7 J=1,N
7     L(I,J)=0
10    DO 8 I=1,N
      D(I,1)=0
      D(I,2)=0
      D(I,3)=0
      Q(I)=0
8     R(I)=0
      Q(S)=1
      D(S,1)=-1
      D(S,2)=S
      D(S,3)=M0
11    K0=0
      DO 9 K=1,N
      IF (Q(K).NE.1) GOTO 9
      DO 21 I=1,N
      IF (R(I)+Q(I).NE.0) GOTO 21
      IF (C(K,I).NE.0.AND.L(K,I).LT.
     # C(K,I)) GOTO 22
      IF (L(I,K).LE.0.OR.C(I,K).EQ.0)
     # GOTO 21
      K0=1
      R(I)=1
      D(I,1)=-1
      D(I,2)=K
      D(I,3)=MIN0(L(I,K),D(K,3))
      GOTO 21
```

```
22      K0= 1
        R(I)= 1
        D(I,1)= 1
        D(I,2)= K
        D(I,3)= MIN0(C(K,I)-L(K,I),D(K,3))
21      CONTINUE
9       CONTINUE
        IF (D(T,1).NE.0) GOTO 12
        IF (K0.EQ.0) GOTO 18
        DO 23 K=1,N
        IF (R(K).EQ.1) Q(K)=1
23      CONTINUE
        GOTO 11
12      DO 24 I=1,N
        P(I)= 0
24      Q(I)= 0
        P(N)=D(T,2)
        K= P(N)
        N1=N-1
        DO 25 I1=1,N1
        I=N-I1
        Q(I)=D(K,1)
        K=D(K,2)
        P(I)=K
        IF (K.EQ.S) GOTO 14
25      CONTINUE
11      K1=P(I)
        DO 26 K=I,N1
        K2=P(K+1)
        IF (Q(K).GT.0) L(K1,K2)=L(K1,K2)
     #  +D(T,3)
```

```
      IF (Q(K).LT.0) L(K2,K1)=L(K2,K1)
   #  +Q(K)*D(T,3)
26    K1=K2
      L(K1,T)=L(K1,T)+D(T,3)
      MC=0
      DO 27 I=1,N
27    MC=MC+L(I,T)
      GOTO 10
13    CONTINUE
      RETURN
      END
```

7.1.5 例　　题

1）用网络最大流算法求解以下问题。

（1）运输网络中，各顶点之间弧上的允许流量与流动方向如图 7.1 所示。求由 1 号顶点出发，接收顶点为 9 号顶点的最大流。

N＝9，M0＝90，S＝1，T＝9

$$
C(I,J)=\begin{vmatrix}
0 & 14 & 0 & 23 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 10 & 9 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 12 & 0 & 0 & 18 & 0\\
0 & 0 & 0 & 0 & 26 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 25 & 4 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 7 & 8 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}
$$

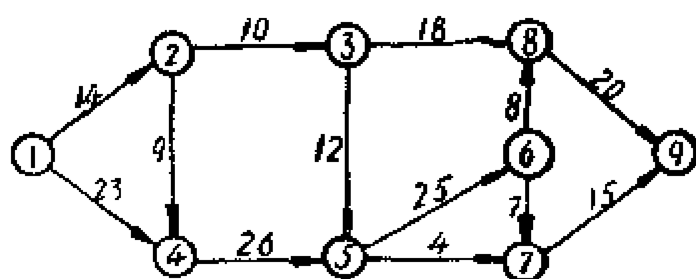


图 7.1

（2）两个煤矿x_1与x_2所产原煤准备以最大的运输量运往三个城市y_1，y_2，y_3。已知这两个煤矿至三个城市的交通运输网的允许流量如图7.2所示。求由x_1，x_2至y_1，y_2，y_3运送原煤的最大值。

首先将这一多出发点多终点的运输流问题变换为一个单出发点单终点的运输流问题来处理。为此引进虚设的出发点S与虚设的终点T。增加五条虚设的有向弧，并给出每条弧上的流量值分别是$\overrightarrow{SX}_1=30$，$\overrightarrow{SX}_2=16$，$\overrightarrow{y_1T}=6$，$\overrightarrow{y_2T}=19$，$\overrightarrow{y_3T}=19$。构成了一个新的流量矩阵C。

$N=12$，$S=1$，$T=12$，流量上界$M0=46$。

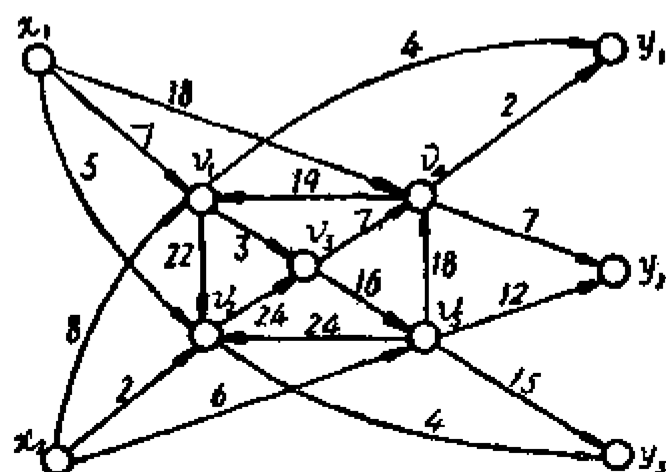


图 7.2

$C(I,J)=$

	S	x_1	x_2	V_1	V_2	V_3	V_4	V_5	y_1	y_2	y_3	T
S	0	30	16	0	0	0	0	0	0	0	0	0
x_1	0	0	0	7	5	0	18	0	0	0	0	0
x_2	0	0	0	8	2	0	0	6	0	0	0	0
V_1	0	0	0	0	22	3	0	0	4	0	0	0
V_2	0	0	0	0	0	24	0	0	0	0	4	0
V_3	0	0	0	0	0	0	7	16	0	0	0	0
V_4	0	0	0	19	0	0	0	0	2	7	0	0
V_5	0	0	0	0	24	0	18	0	0	12	15	0
y_1	0	0	0	0	0	0	0	0	0	0	0	6
y_2	0	0	0	0	0	0	0	0	0	0	0	19
y_3	0	0	0	0	0	0	0	0	0	0	0	19
T	0	0	0	0	0	0	0	0	0	0	0	0

2）计算程序

```
      INTEGER C(30,30),L(30,30)
      WRITE(7,1)
      READ(5,*) N
      WRITE(7,2)
      READ(5,*) M0
1     FORMAT(1X,2HN=)
2     FORMAT(1X,3HM0=)
      CALL MXC(N,M0,C,L)
      STOP
      END
C
      SUBROUTINE MXC (N,M0,G,L)
      INTEGER C(N,N),L(N,N),S,T
      WRITE(7,3)
3     FORMAT(1X,7HC(I,J)=)
```

```
      READ(5,*)((C(I,J),J=1,N),I=1,N)
      WRITE(6,3)
      DO 5 I=1,N
5     WRITE(6,1)(C(I,J,),J=1,N)
1     FORMAT(1X,30I4)
      S= 1
      T=N
      CALL MAXC(N,C,L,S,T,M0,MC)
      WRITE(6,52)
52    FORMAT(1X,7HL(I,J)=)
      DO 53 I=1,N
53    WRITE(6,1)(L(I,J),J=1,N)
      WRITE(6,6)S,T,MC
6     FORMAT(1X,'S=',I2,' T=',I2, 'MC='
    #  ,I5)
      RETURN
      END
```

3）计算结果

（1）MC=29

$$L(I,J)=\begin{bmatrix} 0 & 10 & 0 & 19 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

见图7.3。

（2）MC=39

$L(I,J)=$

	x_1	x_2	V_1	V_2	V_3	V_4	V_5	y_1	y_2	y_3
x_1	0	0	7	5	0	11	0	0	0	0
x_2	0	0	8	2	0	0	6	0	0	0
V_1	0	0	0	10	3	0	0	4	0	0
V_2	0	0	0	0	13	0	0	0	0	4
V_3	0	0	0	0	0	0	16	0	0	0
V_4	0	0	2	0	0	0	0	2	7	0
V_5	0	0	0	0	0	0	0	0	12	10
y_1	0	0	0	0	0	0	0	0	0	0
y_2	0	0	0	0	0	0	0	0	0	0
y_3	0	0	0	0	0	0	0	0	0	0

若以MY_i表示流向Y_i号顶点的最大流量，则$MY_1=6$，$MY_2=19$，$MY_3=14$。

见图7.4。

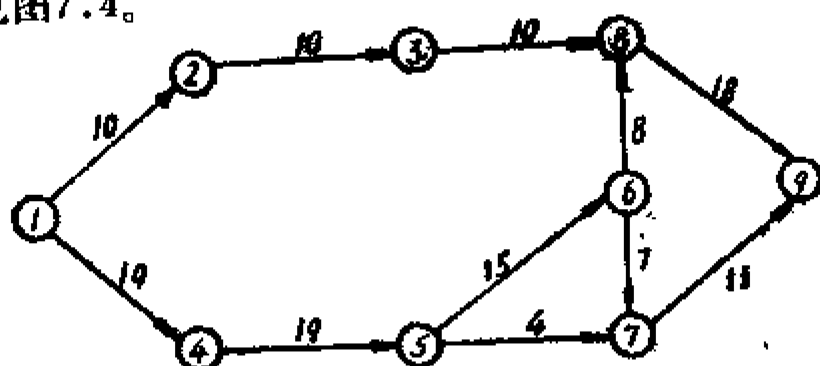


图 7.3

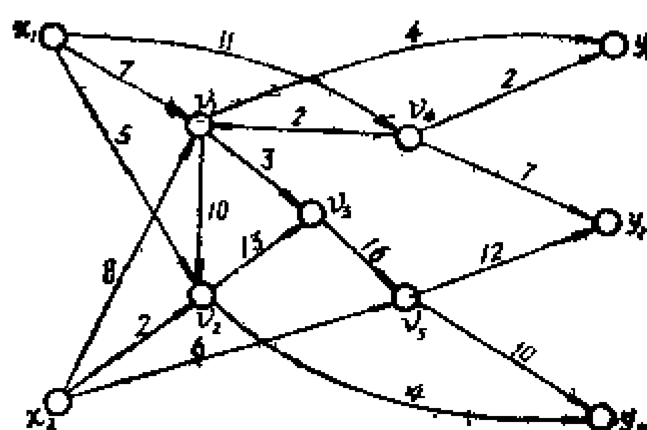


图 7.4

7.2 网络最小费用流的算法

7.2.1 功　能

在运输网络中，每条弧上都有一个最大运输容量值和一个运输费用值（运输单位货物经过这段弧所需要的运费），如何寻找一个由出发点S到终点T的运输方案？该方案的总运输量V是事先给定的，要求这一运输方案在各种使运输量为V的运案中总运费达到极小值。这就是求出发点S到终点T的最小费用流的问题。本节给出一个求最小费用流的计算程序。显然，如果给出的V值大于S点到T点的最大流时，就无法求出运量为V的最小费运流。如果事先不知道S到T的最大流，可以先用前面介绍的最大流算法程序计算最大流，然后再使用计算最小费用流的程序。

7.2.2 方法概述

本节程序中仍然采用了Ford与Fulkerson 提出的算法。它是上一节介绍的网络最大流算法的推广。

首先，设运输网络G中由出发点S到终点T需要运送的货物总量是V。显然，V值不能超过由S至T的最大流$V0$。如果已给定该图的各顶点之间的允许流量矩阵为C，运输费用矩阵为A。图上的顶点X与顶点Y之间的允许流量为$C(X,Y)$，运输费用为$A(X,Y)$。

假定由X至Y的流量为$f(X,Y)$，则最小费用流必须满足下述条件：

$$\sum_{Y}[f(S,Y)-f(Y,S)]=V,$$

$$\sum_{Y}[f(T,Y)-f(Y,T)]=-V,$$

$$\sum_{Y}[f(X,Y)-f(Y,X)]=0$$

对所有的$X \neq S$与　　　$X \neq T$，

$$0 \leqslant f(X,Y) \leqslant C(X,Y)$$

使　　　$$\min\left\{\sum_{(X,Y)} A(X,Y) * f(X,Y)\right\}$$

为了掌握最小费用流的算法，可以先熟悉上一节的网络最大流的Ford与Fulkerson算法。

为了处理的方便，首先给一个一维特征数组P。顶点X的特征数就是数组P的元素$P(X)$。

我们由零流出发。则一开始对图上任何顶点 X 与 Y 有 $f(X,Y)=0$，$P(X)=0$。设法按照运输费用最省的路线增加流量，并根据需要去增加特征数$P(X)$的值，从而重新得到增加流量的机会。只要总流量值V不超过由S至T的最大流量$V0$，都可以找到费用最省、流量为V的运输方案。

应当指出的是，本算法中为了找到一条增广路，除了按最大流算法那样去找前进弧与背向弧的条件外，还需要增加判断条件：

$$P(Y)-P(X)=A(X,Y)$$

如果在这样的弧上仍然找不到增广路，则应当使未标号顶点X的特征数$P(X)$的值加1，再去进行判断与寻找。

在计算过程中有两种转出迭代的可能。一种是当流量值已达到给定的V值，已经找出了最小费用流，结束计算。另一种情况是，当未标号顶点Y的特征数 $P(Y)$与已标号顶点X的特征数 $P(X)$之差均超过了本题所给的每条对应弧的费用值时，也就不必继续迭代下去。因为再也找不到一条可增

加运输流量的由 S 至 T 的增广路了。这样结束迭代时，得到的是总流量值小于给定总流量 V 的最小费用流。

7.2.3 子程序参数说明

子程序名称 MINFC(N,C,A,L,S,T,V0,V)

N：图的顶点数目，整型变量。

C(N,N)：图上各顶点间的弧的允许流量输入矩阵，整型数组。

A(N,N)：图上各弧的运输费用输入矩阵，整型数组。注意，当 $C(i,j)=0$ 时，对应的费用值 $A(i,j)$ 也取零值，这些数组分量在计算过程中不参与计算。

L(N,N)：最小费用流的流量输出矩阵，整型数组。

S：运输流的出发点序号，整型变量。

T：运输流的终点序号，整型变量。

V0：指定由 S 至 T 的运输流量的输入数据，它不应当大于由 S 至 T 的最大流量值，整型变量。

V：最小费用流的流量值输出数据，整型变量。如果 $V0$ 的取值不超过最大流量值，则有 $V=V0$，如果 $V0$ 取值超过最大流的值，则 V 的结果取 S 至 T 的最大流值。

7.2.4 子 程 序

```
      SUBROUTINE MINFC(N,C,A,L,S,T,V0,V)
      INTEGER V0,V,S,T,C(N,N),A(N,N,),L(N
   #  ,N),D(30,3),P(30),Q(30),R(30)
      V=0
      KK=V0
```

```
      KA=0
      DO 2 I=1,N
      P(I)=0
      DO 1 J=1,N
      L(I,J)=0
1     KA=MAX0(KA,A(I,J))
2     CONTINUE
11    DO 3 I=1,N
      D(I,1)=0
      D(I,2)=0
      D(I,3)=0
      Q(I)=0
3     R(I)=0
      D(S,1)=-1
      D(S,2)=S
      D(S,3)=V0
      Q(S)=1
      K0=0
12    K1=0
      DO 9 K=1,N
      IF (Q(K).NE.1) GOTO 9
      DO 21 I=1,N
      IF (R(I)+Q(I).NE.0) GOTO 21
      IF (C(K,I).NE.0.AND.L(K,I).LT.C(K,I)
   #  .AND.P(I)-P(K).EQ.A(K,I)) GOTO
   #  22
      IF (L(I,K).LE.0.OR.C(I,K).EQ.0.OR.
   #  P(K)-P(I).NE.A(I,K)) GOTO 21
      K1=1
      R(I)=1
      D(I,1)=-1
```

```
      D(I,2)=K
      D(I,3)=MIN0(L(I,K),D(K,3))
      GOTO 21
22    K1=1
      R(I)=1
      D(I,1)=1
      D(I,2)=K
      D(I,3)=MIN0(C(K,I)-L(K,I),D(K,3))
21    CONTINUE
9     CONTINUE
      IF (K1.NE.1) GOTO 24
      DO 23 I=1,N
      IF (R(I).EQ.1) Q(I)=1
23    CONTINUE
      IF (D(T,1).EQ.0) GOTO 12
24    K0=K0+1
      IF (D(T,1).NE.0) GOTO 13
      IF (K0.GT.KA+1) GOTO 10
      DO 35 I=1,N
      IF (R(I)+Q(I).EQ.0) P(I)=P(I)+1
35    CONTINUE
      GOTO 12
13    D(T,3)=MIN0(D(T,3),KK)
      KK=KK-D(T,3)
      DO 25 I=1,N
      R(I)=0
25    Q(I)=0
      R(N)=D(T,2)
      K=D(T,2)
      N1=N-1
      DO 26 I1=1,N1
```

```
          I = N - I1
          Q(I) = D(K,1)
          K = D(K,2)
          R(I) = K
          IF (K.EQ.S) GOTO 14
26        CONTINUE
14        K1 = R(I)
          DO 28 K = 1,N1
          K2 = R(K+1)
          IF (Q(K).GT.0) L(K1,K2) = L(K1,K2)+
       #  D(T,3)
          IF (Q(K).LE.0) L(K2,K1) = L(K2,K1)-
       #  D(T,3)
27        K1 = K2
28        CONTINUE
          L(K1,T) = L(K1,T)+D(T,3)
          V = 0
          DO 29 I = 1,N
29        V = V+L(I,T)
          IF (KK.GT.0) GOTO 11
10        CONTINUE
          RETURN
          END
```

7.2.5 例　　题

1）七个顶点的运输网络的邻接关系如图 7.5 所示。每条弧上都标有两个数字。第一个数字是该弧上单位货物的运费，第二个数字是这条弧的最大运输容量。试求出发点 $S=1$ 终点 $T=7$ 时，总运输量为38与总运输量为20时的网个最

小费用运输流的方案。

$$A(I,J)=\begin{pmatrix} 0 & 7 & 0 & 13 & 0 & 28 & 0 \\ 0 & 0 & 25 & 4 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 6 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 3 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C(I,J)=\begin{pmatrix} 0 & 16 & 0 & 11 & 0 & 13 & 0 \\ 0 & 0 & 17 & 18 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 22 \\ 0 & 0 & 12 & 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \\ 0 & 0 & 0 & 0 & 10 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

（1）N=7，S=1，T=7，V0=38。

（2）N=7，S=1，T=7，V0=20。

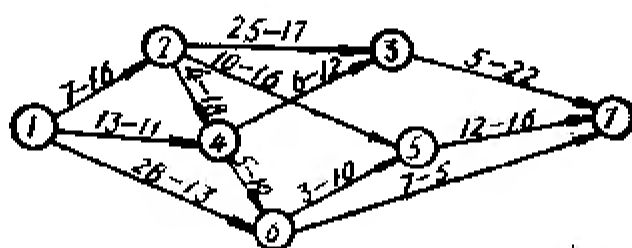


图 7.5

2）计算程序

```
INTEGER C(30,30), A(30,30), L(30,30), V0
WRITE(7,1)
READ(5,*)N
```

```
      WRITE(7,2)
      READ(5,*)V0
1     FORMAT(1X,2HN=)
2     FORMAT(1X,3HV0=)
      CALL MINC(N,V0,C,A,L)
      STOP
      END
C
      SUBROUTINE MINC(N,V0,C,A,L)
      INTEGER V0,C(N,N),A(N,N),L(N,N),
     # S,T,V
      WRITE(7,3)
3     FORMAT(1X,7HA(I,J)=)
      READ(5,*)(A(I,J),J=1,N),I=1,N)
      WRITE(7,4)
4     FORMAT(1X,7HC(I,J)=)
      READ(5,*)(C(I,J),J=1,N),I=1,N)
      S=1
      T=N
      CALL MINFC(N,C,A,L,S,T,V0,V)
      WRITE(6,7)S,T,V
7     FORMAT(1X,'S=',I2,' T=',I2,
     #         V=',I6)
      DO 8 I=1,N
8     WRITE(6,1)(L(I,J),J=1,N)
1     FORMAT(1X,30I5)
      RETURN
      END
```

3）计算结果

（1）当V0=38时，见图7.6。

V = 38

$$L(I,J)=\begin{vmatrix} 0 & 16 & 0 & 11 & 0 & 11 & 0 \\ 0 & 0 & 5 & 1 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 17 \\ 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 \\ 0 & 0 & 0 & 0 & 6 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

（2）当V0＝20时，见图7.7。

V = 20

$$L(I,J)=\begin{vmatrix} 0 & 16 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 12 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

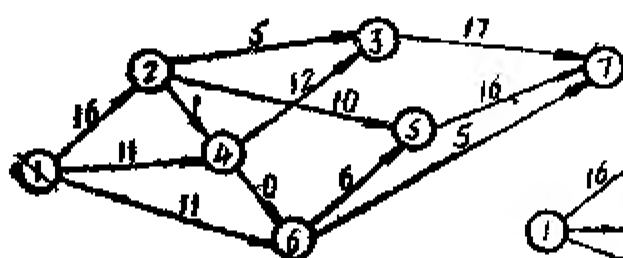


图 7.6

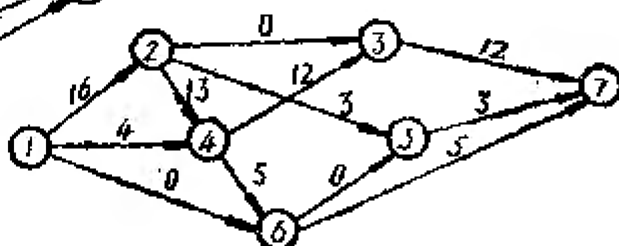


图 7.7

主要参考资料

1. 《图论及其应用》 卢开澄著 清华大学出版社 1981年
2. 《图论》 王朝瑞编 人民教育出版社 1981年
3. 《有趣的图论》 管梅谷 山东科学技术出版社 1980年